UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Motor de Aventuras Gráficas tipo SCUMM

Estudiante: Adrián Granero Tierno
Tutor: Carlos Aguirre Maeso

Febrero 2020

Motor de Aventuras Gráficas tipo SCUMM

AUTOR: Adrián Granero Tierno
TUTOR: Carlos Aguirre Maeso

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero de 2020

Resumen (castellano)

A partir del motor gráfico Unity 3D se tiene la capacidad para desarrollar videojuegos sin requerir un alto grado de programación y si se quiere crear una aventura gráfica hay que implementar las funcionalidades desde cero o usar las que ofrece Unity en su tienda siendo la mayoría de ellas de pago o de calidad no muy alta.

Sin embargo, si no se quiere usar Unity, el usuario ha de buscar un motor gráfico únicamente enfocado en generar aventuras gráficas tipo SCUMM. Estos motores gráficos podrían no adecuarse a lo que el usuario busca, por falta de funcionalidades o requerir un nivel de programación alto.

Este Trabajo Fin de Grado ayudará al usuario a crear y desarrollar una aventura gráfica en 2D a partir de herramientas y elementos de la interfaz Unity.

Para ello, se ha procedido a crear e implementar scripts y prefabs con los cuales se cubrirán las necesidades básicas que tiene un usuario para crear los elementos de una aventura gráfica. Estas necesidades pueden ser la de crear un sistema de diálogos entre personajes, añadir zonas de interacción para el jugador con sus posibles eventos y funcionalidades o la creación y eliminación de objetos con su correspondiente sistema de inventario. Todo ello estará desarrollado en el lenguaje de programación C# (C Sharp) con el programa Microsoft Visual Studio.

A continuación, se tratarán los principales sistemas los principales sistemas que se han desarrollado:

Empezando por el sistema de interacciones con el cual los NPC’s, cambios de escenas y objetos generarán cierta interacción con el jugador ya sea proporcionando información, nuevas zonas de exploración o elementos que puedan provocar un evento.

El siguiente sistema se enfoca en los objetos y el inventario permitiendo la recogida de elementos y su almacenaje. Además, amplificará las funcionalidades del sistema de interacciones que tiene el juego.

El conjunto de diálogos es uno de los sistemas creados, ya que el poder hablar con personajes sirve para entender y obtener información necesaria para llegar a un objetivo en concreto.

Por último, tenemos los eventos que, a partir del uso de los sistemas anteriores, pueden ofrecer la posibilidad de provocar una acción, lo cual puede desbloquear elementos ocultos o nuevas acciones.

En un principio estas son las características principales que posee este conjunto de Assets, con ellas se ha implementado una pequeña demo que permitirá ver la capacidad para generar una aventura gráfica.

Palabras clave (castellano)

Unity, 2D, aventura gráfica, Microsoft Visual Studio, motor gráfico, C Sharp, Assets, NPC’s, SCUMM.

Abstract (English)

From the Unity 3D graphic engine you have the capacity to develop video games without requiring a high degree of programming and if you want to create a graphic adventure you have to implement the functionalities from scratch or use those offered by Unity in your store most of them being paid or not very high quality.

However, if you don't want to use Unity, the user has to look for a graphic engine solely focused on generating SCUMM type graphic adventures. These graphic engines might not fit what the user is looking for, lack of functionalities or require a high level of programming.

This End of Degree Project will give the user the facility to create and develop a 2D graphic adventure from tools and elements of the Unity interface.

For this purpose, scripts and prefabs were created to cover the basic needs that a user has to create the elements of a graphic adventure. These needs can be the creation of a system of dialogues between characters, an addiction of interaction zones for the player with their possible events and functionalities or the creation and elimination of objects with their corresponding inventory system. All this will be developed in the programming language C# (C Sharp) with the program Microsoft Visual Studio.

The main systems that have been developed are these:

Starting with the system of interactions with which the NPC's, scene changes and objects will generate some interaction with the player either by providing information, new zones of exploration or elements that may cause an event.

The next system focuses on objects and inventory, allowing the collection of elements and their storage. In addition, it will amplify the features of the interaction system that the game has.

The set of dialogues is one of the systems created, since, being able to talk to characters serves to understand and obtain new information getting to a specific objective.

Finally, we have the events that from the use of the previous systems can offer the possibility of provoking an action, which can unlock hidden elements or new actions.

At the beginning these are the main characteristics that this set of Assets, with them a small demo has been implemented showing the capacity to generate a graphic adventure.

Keywords (inglés)

Unity, 2D, graphic adventure, Microsoft Visual Studio, graphic engine, C Sharp, Assets, NPC’s, SCUMM.

Agradecimientos

Llegar al final de la carrera y de este TFG no hubiera sido posible sin el apoyo y esfuerzo por parte de mi familia, sobre todo de mis padres y mi hermana, es por ello que les quiero dar las gracias.

Me gustaría dar las gracias a mis amigos y compañeros de carrera con los cuales he superado todas las prácticas de las asignaturas.

Por último, quiero dar las gracias a los profesores que me han aconsejado y ayudado a lo largo de estos años y también a Carlos Aguirre, mi tutor de TFG, que me ha estado ayudando en las asignaturas que tuve con él y en este TFG.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

El mundo de los videojuegos de aventuras gráficas es muy amplio, con diversas mecánicas de juego, entornos en 2D o 3D y varios subgéneros. Además, este tipo de juegos sigue casi siempre una estructura formada por unos conjuntos de elementos en común.

Por otro lado, Unity, en su conjunto, proporciona una interfaz gráfica sencilla con capacidad de implementación de elementos, modificación de estos, animación de objetos, etc. Asimismo, Proporciona una posibilidad de creación de scripts para dar forma a los objetos que se ha mencionado anteriormente, realización de eventos o diagramas de flujo para desarrollar una animación.

Por lo que se puede observar, Unity ofrece muchas herramientas y posibilidades de creación. Sin embargo, la falta de conocimientos suficientes de programación y de las herramientas que se proporcionan implica no poder aprovechar todo el potencial del motor Unity.

Cabe destacar que se puede hacer uso de la tienda de Assets, la cual ofrece una gran variedad de elementos y funcionalidades ya creadas. En cambio, si se buscan elementos en específico para desarrollar una aventura gráfica, estos se centran en una única funcionalidad. Por lo general hay muy poco material y el que existe está relacionado con otro tipo de juegos o géneros.

Entre los Assets más destacados con relación a una aventura gráfica se encuentran los siguientes:

- **«AdventureCreator»:** es uno de los Assets más completos en cuanto a funcionalidades, pero su mayor inconveniente recae en la dificultad de adaptación con otros proyectos o Assets.

  https://assetstore.unity.com/packages/templates/systems/adventure-creator-11896

- **«Adventure-SampleGame»:** es un Asset creado por los desarrolladores de Unity y cuenta con tutoriales y videos explicativos para entender su uso. No obstante, existen ciertas incompatibilidades con algunas versiones de Unity y múltiples errores de compilación.

  https://assetstore.unity.com/packages/essentials/tutorial-projects/adventure-sample-game-76216

- **«Game Creator»:** con este Asset se pueden crear funcionalidades simples, pero si se desea crear algo más complejo hay que comprar sus módulos adicionales.

  https://assetstore.unity.com/packages/templates/systems/game-creator-89443

Debido a la escasez de herramientas en conjunto, este Trabajo Fin de Grado busca dar la posibilidad de generar una aventura gráfica con Unity desde cero en base a elementos que pueden ser configurados por el usuario.

1.2 Objetivos

El objetivo principal del TFG es poder crear un motor tipo SCUMM con las herramientas necesarias para que un usuario con o sin conocimientos sobre programación en C# u otro lenguaje pueda crear una aventura gráfica 2D en Unity.

Para la realización de estas herramientas se hará uso del motor gráfico de Unity 3D y del lenguaje de programación C# con el programa Microsoft Visual Studio.

El género de videojuegos al que va dirigido este proyecto es el de las aventuras gráficas, en concreto se enfoca en las dos dimensiones (2D), consiguiendo con esto que el juego que se cree tenga una jugabilidad clásica de las aventuras gráficas. Asimismo, las herramientas que se implementan adaptarán las mecánicas de estos juegos.

A parte del desarrollo de las herramientas, se procederá a la explicación del uso de cada una de ellas, el propio funcionamiento de estas entre sí y la implementación que se llevó a cabo.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos y anexos:

- **Capítulos:** Aquí se procederá a explicar todo lo relacionado con la historia de las aventuras gráficas, motores centrados en juegos de aventuras gráficas, el desarrollo del motor gráfico de Unity 3D y, por otro lado, se describirá la implementación, diseño y desarrollo de las herramientas creadas en este proyecto.

    1. **Introducción:** En este capítulo daremos una breve explicación de la motivación para llevar a cabo este TFG y a su vez describiremos sus objetivos y la organización que se seguirá.

    2. **Estado del Arte:** En este capítulo se describe la historia de las aventuras gráficas, algunos de los motores dedicados a crear aventuras gráficas y el motor gráfico Unity 3D.

    3. **Diseño:** En este capítulo se dará un análisis del diseño seguido indicando los patrones usados y los diagramas de clases de las herramientas creadas.

    4. **Desarrollo:** En este capítulo se explicará la creación de los Assets y la implementación entre ellos.

    5. **Integración, pruebas y resultados**: En este capítulo se procederá a ver la integración llevada a cabo en Unity, las pruebas que se realizaron y los resultados de estas.

6. **Conclusiones y trabajo a futuro:** En este último capítulo se hablará de las conclusiones obtenidas al realizar este trabajo fin de grado y de lo que se puede ampliar si se continúa en el futuro.

- **Anexos:**

A. **Manual de Instalación:** Se indican las instrucciones para instalar o importar el Asset del proyecto a través de capturas de pantalla y breves descripciones.

B. **Manual del Asset:** Se indican las carpetas que contiene el Asset y sus archivos internos. También se añaden explicaciones de los prefabs creados, el cómo usarlos y capturas de pantalla para una mejor comprensión.

C. **Anexo Imágenes de los test:** Se muestran capturas de los test realizados en la pequeña demo creada.

2 Estado del arte

2.1 Videojuegos de aventuras gráficas

Los videojuegos surgieron en el año 1970 con la aparición de las primeras máquinas recreativas. Un año más tarde, en 1971, Nolan Bushnell empezó a comercializar Computer Space, el cual es un videojuego con características similares a los Space War. Tras esto, en el año 1972, llegó el juego de tenis que supuso la base del videojuego como industria, conocido como Pong.

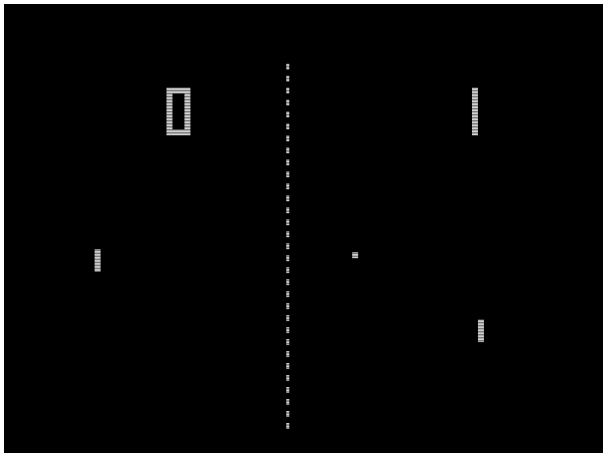

Figura 1 de Estado del arte: Juego Pong

Con el fin de los años 70, nació en Estados Unidos y en Inglaterra el género de aventura conversacional. Este tipo de juegos quería dar al jugador un tipo de aventuras a partir de diálogos, en los cuales se escriben ciertas órdenes por teclado y provocan una respuesta por parte del juego. Estas aventuras conversacionales precedieron al género de las aventuras gráficas.

Figura 2 de Estado del arte: Aventura conversacional

A medida que avanzaba el tiempo aparecían ordenadores más potentes, nuevas tecnologías y gráficos. Debido a estos avances, el género de las aventuras gráficas nació con la aparición del juego llamado «King's Quest: Quest for the Crown» en el año 1984. Este juego fue la primera aventura en la que sen podía interactuar con los gráficos.

Figura 3 de Estado del arte: Juego King's Quest: Quest for the Crown

Finalmente, se hablará de una de las sagas de aventuras gráficas más representativas de este género, se trata de «The Secret of Monkey Island».

El primer videojuego de esta saga fue publicado en octubre de 1990. Tuvo un gran impacto y debido a su éxito se crearon cuatro secuelas:

- Monkey Island 2: LeChuck's Revenge
- The Curse of Monkey Island
- Escape from Monkey Island
- Tales of Monkey Island.

La jugabilidad de estas aventuras gráficas tenía un sistema clásico y hacía uso del point and clic, en el que se podía mover al personaje con el ratón, se podían escoger opciones y/o objetos disponibles en el menú y además había interacción con personajes y elementos mostrados en la pantalla.

Figura 4 de Estado del arte: Monkey Island

2.2 Motores de aventuras gráficas

Estos son algunos de los programas empleados para desarrollar juegos enfocados en el género de las aventuras gráficas:

2.2.1 Adventure Game Studio (AGS):

Fue creado por Chris Jones para poder desarrollar aventuras gráficas. Se trata de un programa de código abierto con una interfaz gráfica y un editor para realizar scripting en el lenguaje de programación C.

Algunas de sus principales capacidades:

- Está diseñado para funcionar en varios sistemas operativos como Microsoft Windows, Android, iOS, Linux, Mac OS X y PSP.
- Admite varios filtros gráficos y formatos multimedia.
- Tiene una gran comunidad de usuarios divididos en foros, blogs y canales (AGS Forum, AGS Internet Realy Chat y Discord).

Algunos de sus inconvenientes técnicos:

- Un aumento de la resolución gráfica implica una mayor potencia por parte del ordenador.
- Respecto a sus competidores, los juegos de AGS requieren más ocupación de recursos.
- Limitación técnica si no se tienen conocimientos de scripting.

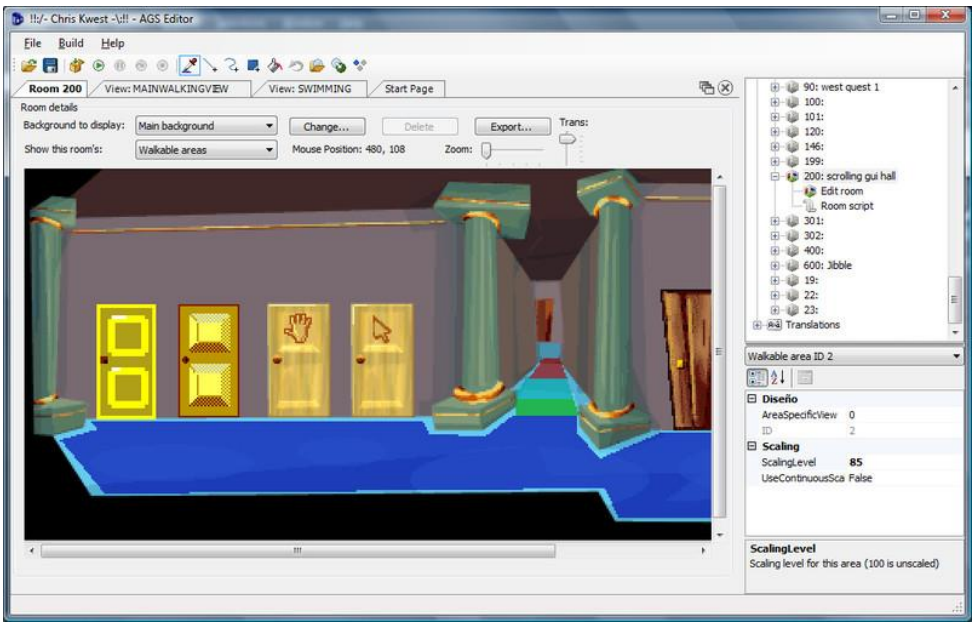

Figura 5 de Estado del arte: Adventure Game Studio (AGS)

2.2.2 WinterMute Engine

Fue creado por Jan Nedoma y se trata de un conjunto de herramientas con un intérprete de guiones para poder implementar la lógica del juego. Se construyó como un motor gráfico de aventuras 2D con posible combinación de personajes en 3D dando así gráficos en 2.5D.

Algunas de sus principales capacidades:

- Uso de la aceleración 3D para lograr gráficos 2D en altas resoluciones.
- Variedad de formatos de archivos admitidos.
- Lenguaje de scripting en C.
- Libertad de uso en cuanto a resoluciones, colores y sonidos.

Algunos de sus inconvenientes técnicos:

- Los juegos 2D solo funciona con emulación en los sistemas operativos de Linux y Mac.
- Problemas en la comprobación de la validez de las licencias.
- Limitación técnica si no se tienen conocimientos de scripting.

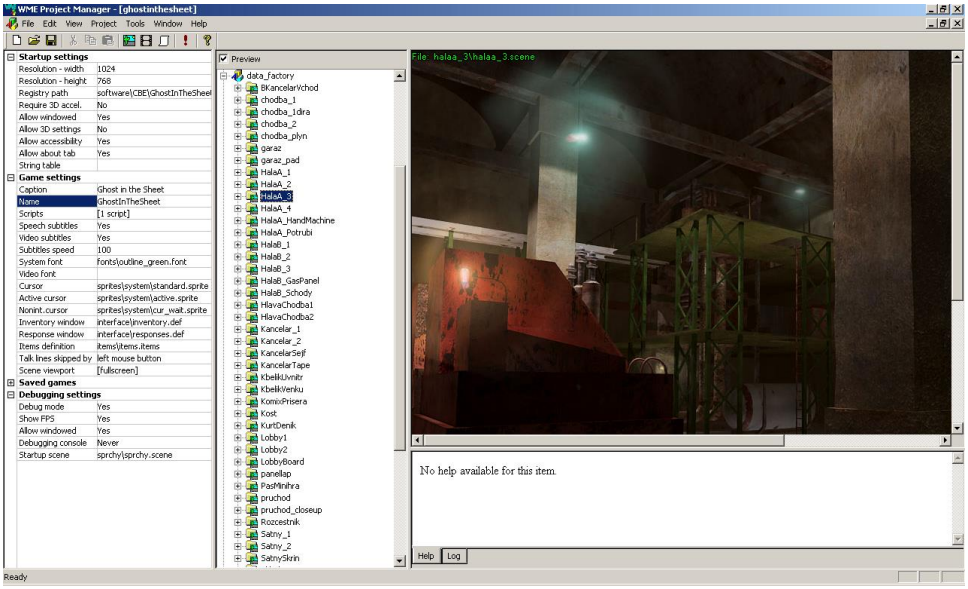

Figura 6 de Estado del arte: WinterMute Engine

2.2.3 ScummVm

Fue creado por Ludvig Strigeus para poder crear juegos de aventuras y emular los juegos en el motor SCUMM. Es una herramienta de creación de scripts con un conjunto de recreaciones de motores de juegos.

Algunas de sus principales capacidades:

- Soporte para los desarrolladores al firmar acuerdos favorables como por ejemplo recibir código fuente, asesoramiento técnico y activos de trabajo.
- Compatibilidad con otros motores gráficos como Nailhead AGI Studio, Gob, AGI (Adventure Game Interpreter), entre otros.
- Está disponible en varias plataformas como Microsoft Windows, Linux, iOS, Linux, Mac OS, Xbox, Android, PlayStation 2, Nintendo DS, entre otros.

Algunos de sus inconvenientes técnicos:

- Al estar enfocado en la creación de scripts se necesita tener ciertos conocimientos para crear juegos nuevos y originales.
- La mayoría de los juegos que se crean siguen un diseño verbo-objeto (personaje con un inventario, muchos objetos e interacción con verbos para realizar las acciones), esto puede ser un limitante si se busca crear otro estilo de aventuras gráficas.

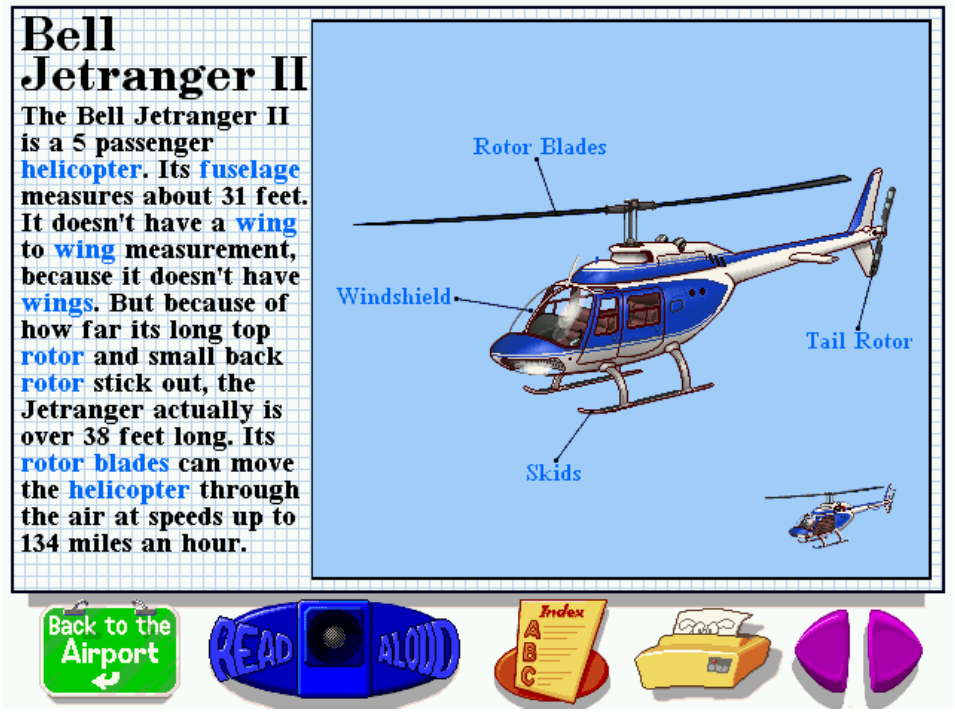

Figura 7 de Estado del arte: ScummVm

2.2.4 Visionaire Studio

Es un motor gráfico con la capacidad de hacer muchas variantes de aventuras gráficas.

Algunas de sus principales capacidades:

- No es necesario saber programar gracias a su interfaz de usuario intuitiva y su modelo de diseño.
- Tienen diversas licencias para cada tipo de desarrollo.
- Interfaz gráfica (OpenGL y OpenGL ES) y de audio (OpenAL).
- Programa multilenguaje (alemán, inglés, francés, holandés, español e italiano).
- Es multiplataforma (Microsoft Windows, Linux, iOS, Linux, Mac OS, Xbox One, Android, PlayStation 4).

Algunos de sus inconvenientes técnicos:

- No tiene licencia gratuita.
- No es recomendable si se planea usar pixel art.
- Está dirigido a juegos en alta resolución.

Figura 8 de Estado del arte: Visionaire Studio

2.3 Motor gráfico Unity

Descripción:

Unity se trata de un motor de videojuegos tanto 2D como 3D, el cual permite crear y desarrollar videojuegos con gráficos actuales debido a la gran potencia que tiene y las actualizaciones que ha ido recibiendo a lo largo de los años.

Este motor de videojuego va dirigido tanto a las pequeñas y grandes empresas por igual, ofreciendo su programa a empresas, estudiantes universitarios y a particulares.

El entorno en el que se trabaja con Unity es bastante sencillo y familiar tanto si lo usa un programador, un diseñador o un estudiante. Además, es multiplataforma por lo que puede ejecutarse en distintos sistemas operativos.

En suma, Unity cuenta con un gran número de características y funcionalidades debido a su gran variedad de herramientas, lo cual permite desarrollar diversos videojuegos.

Figura 9 de Estado del arte: Logo de Unity

Las versiones más importantes de Unity son las siguientes:

- En el año 2005 salió la primera versión de Unity en la Conferencia Mundial de Desarrolladores de Apple. Sus creadores enfocaron el funcionamiento de Unity exclusivamente para la plataforma de ordenadores Mac. Esta versión inicial obtuvo las ganancias suficientes para poder continuar desarrollándolo.

- Después de 5 años (en septiembre del año 2010), se presentó Unity 3 y esta nueva versión se centró sobre todo en atraer a grandes desarrolladores de videojuegos, ofreciendo herramientas más potentes tanto a grandes, pequeñas e independientes desarrolladoras a un precio más asequible.

- Y, por último, se encuentra Unity 5. Fue lanzado y presentado a inicios del año 2015, más concretamente el 3 de marzo. Esta versión fue anunciada en la Game Developers. Entre sus añadidos tenemos un sistema de animación, llamado Mecanim animation, soporte para DirectX 1, soporte para juegos en Linux y arreglo de bugs y texturas.

Características:

- Una de las características de las que se han mencionado anteriormente, es el poder desarrollar un videojuego en 2D como en 3D dándole al usuario tanto un gran poder creativo, como una gran variedad de posibilidades para los diversos géneros que existen.

- La posibilidad de usar los prefabs que proporciona Unity, es una de las características más empleadas, ya que permite crear flujos de trabajo más eficientes y flexibles. Estos prefabs o GameObjects pueden o bien estar ya configurados o se pueden configurar de una forma sencilla, gracias a estos el usuario puede ahorrarse bastante tiempo.

- Creación sencilla y rápida de interfaces de usuarios a través de un sistema UI incorporado en el motor gráfico.

- Unity incorpora PhysX con NVIDIA como motor de físicas para crear juegos más realistas, potentes y llenos de detalles. En cuanto a las físicas en juegos 2D utiliza Box2, usando una tecnología basada en datos, llamada DOTS (Data-Oriented Technology Stack).

- En la Asset Store se pueden crear y agregar herramientas personalizadas a disposición de los usuarios dando la facilidad de poder expandir la capacidad de creación de Unity. Estas herramientas, recursos y mecanismos ofrecen proyectos más ágiles y accesibles a los usuarios.

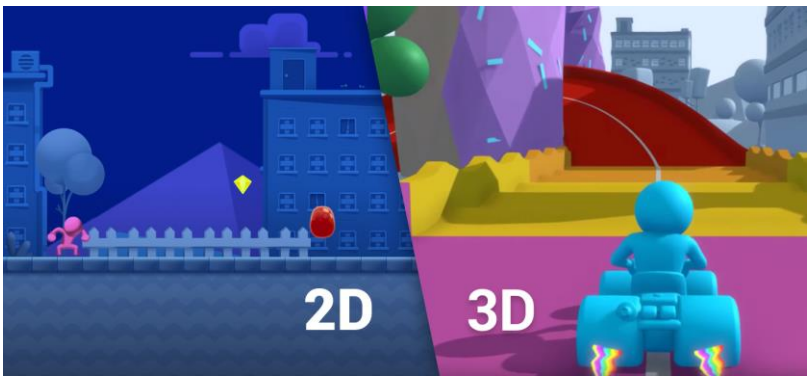

Figura 10 de Estado del arte: Entorno 2D y 3D

Multiplataforma:

Unity es un motor gráfico que puede ser usado en distintas plataformas.

Proporciona un entorno sencillo y fácil de usar para llegar a todas las plataformas más populares, sin obligar a los usuarios a programar el juego específicamente para cada una de ellas.

Unity está disponible en ordenadores de sobremesas (Windows, Mac y Linux), plataformas móviles (iOS, Android y BlackBerry), consolas (PlayStation, Xbox, Wii, Wii U, …) y en páginas Web.

3 Diseño

Durante esta sección se especificarán las decisiones de diseño tomadas, el diseño de los Assets, los métodos que se han usado y las relaciones entre los Assets mediante un diagrama de clases.

3.1 Decisiones de diseño

Se tomaron ciertas consideraciones antes de empezar a realizar este proyecto, las cuales se centraron en encontrar cuatro o cinco pilares interconectados entre sí para generar una aventura gráfica.

Debido a esto, el proyecto está dividido en cuatro partes o sistemas que son los diálogos, las interacciones, el inventario y los eventos.

3.2 Patrones de diseños

Patrones de diseño utilizados en el TFG:

- **Constructor:** Se crean varios objetos simples para formar un objeto complejo, pudiendo añadir a este nuevas características o funcionalidades.
  Principalmente usado en el sistema de creación de diálogos.

- **Singleton:** Se crean objetos para las clases pertenecientes al sistema de inventario. Con ello se asegura la creación de una única instancia global permitiendo a las demás clases acceder a los datos disponibles en ella.
  Se usa en la clase Inventory, creando su instancia y pudiendo añadir o eliminar objetos a una lista.

- **Prototype:** Para instanciar muchos elementos u componentes iguales o clónicos con el mismo script, se usa este patrón de Unity para la creación de clones de un componente con un script en común.
  Se usa con el script de InventorySlot y el objeto Slot.

3.3 Diagrama de clases

Como se decidió crear cuatro sistemas o implementaciones centrados en una característica de las aventuras gráficas, se mostrarán los diagramas de clases de cada uno por separado.

En estos diagramas de clase se pueden observar los scripts, sus parámetros y sus funciones.

3.3.1 Diagrama de clases del sistema de diálogos

Este sistema se decidió dividir en dos partes:

Dialog Manager: Se encarga de manejar toda la parte de interpretación de las claves, analizar los textos, la transición entre los diálogos y además controla la pulsación de los botones de los diálogos.

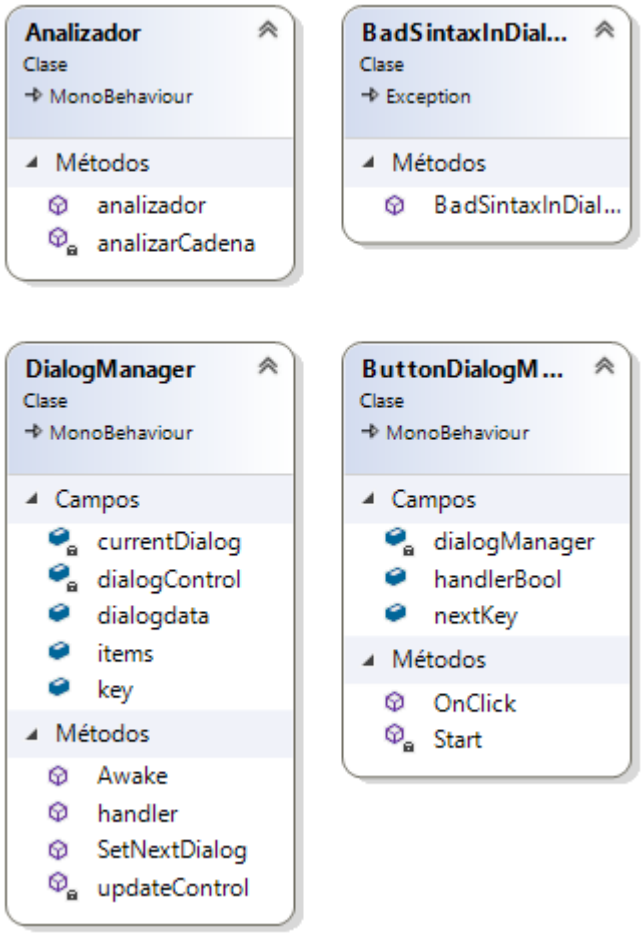

Figura 1 de Diseño: Diagrama de clases diálogo

Dialog System: Se encarga de toda la creación de los archivos de diálogos, la interfaz GUI para editar los textos y claves, y de la declaración de los contructores de los diálogos y de los botones.

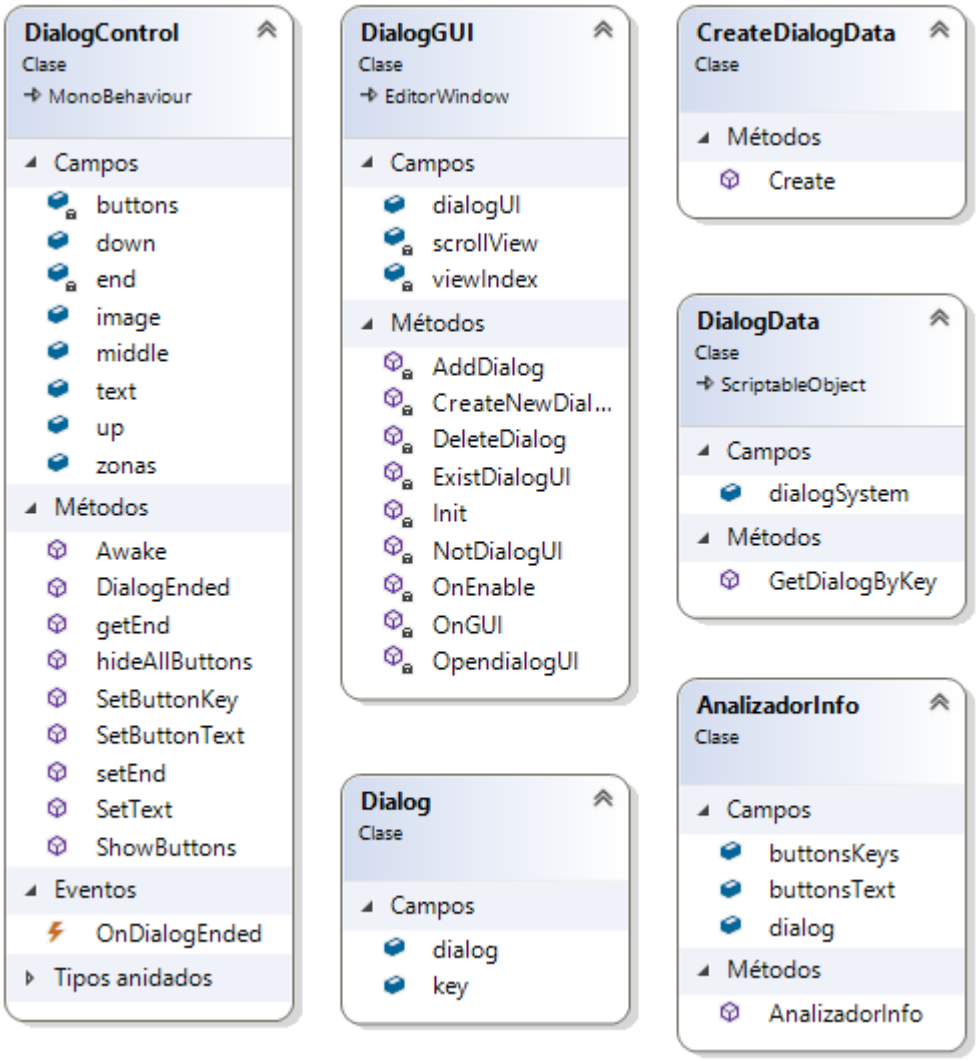

Figura 2 de Diseño: Diagrama de clases diálogo 2

3.3.2 Diagrama de clases del sistema de interacciones

Las interacciones están formadas por tres scripts, los cuales pueden funcionar independientemente o en conjunto.

Zona Interacción: Se encarga de los diálogos u objetos, realizando la llamada a estos cuando se produce una acción.

Cambiar Escenas: Se encarga de proporcionar la acción de deshabilitar la escena actual y habilitar la nueva escena que tiene indicada.

Escena Interacción: Se encarga de proporcionar un diálogo con asociación a un evento cuando se intenta producir un cambio de escena.

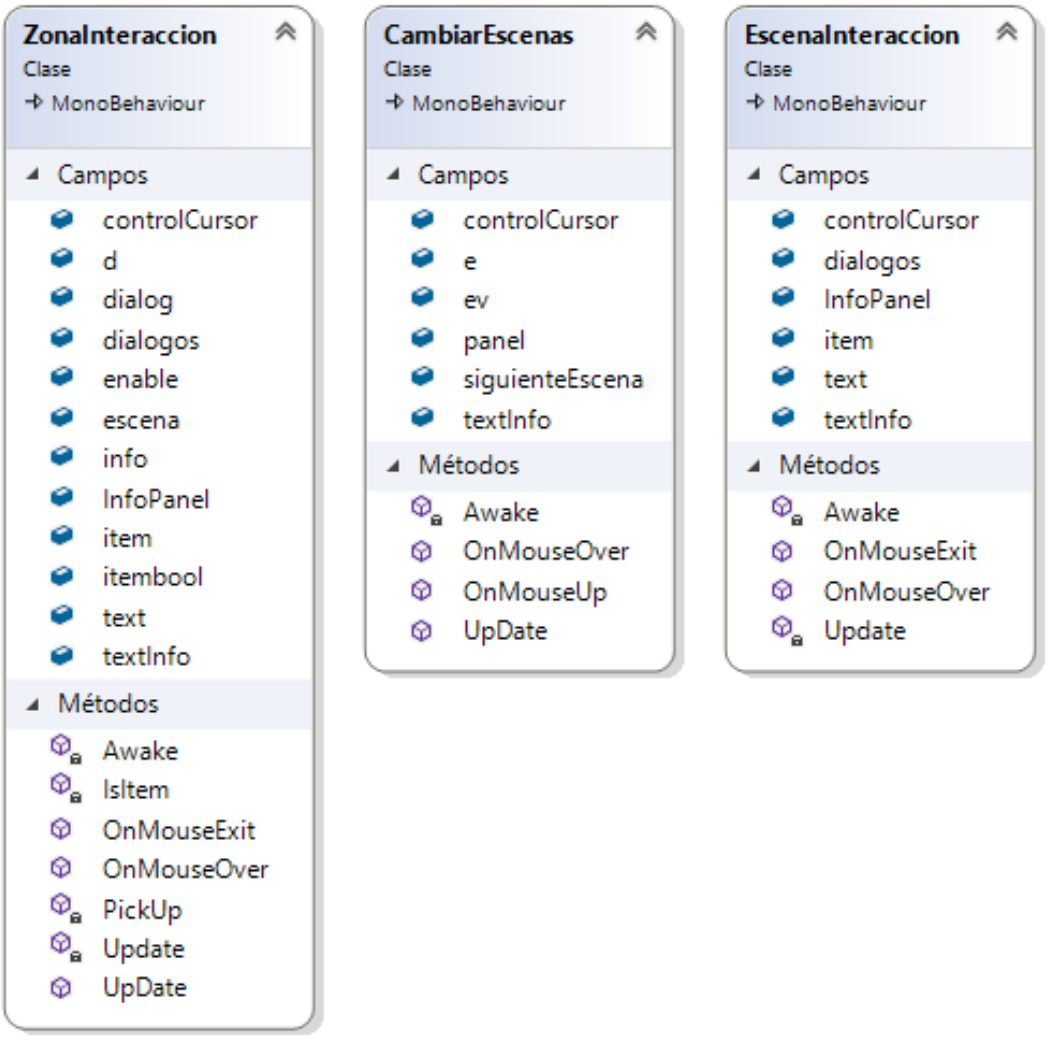

Figura 3 de Diseño: Diagrama de clases Interacción

3.3.3 Diagrama de clases del sistema de inventario

El inventario y los objetos están formados por sus constructores y la interfaz que se muestra durante la ejecución del juego.

La clase Inventory es la que usa el patrón Singleton para proporcionar un único inventario con una lista limitada de objetos que pueden ser recogidos.

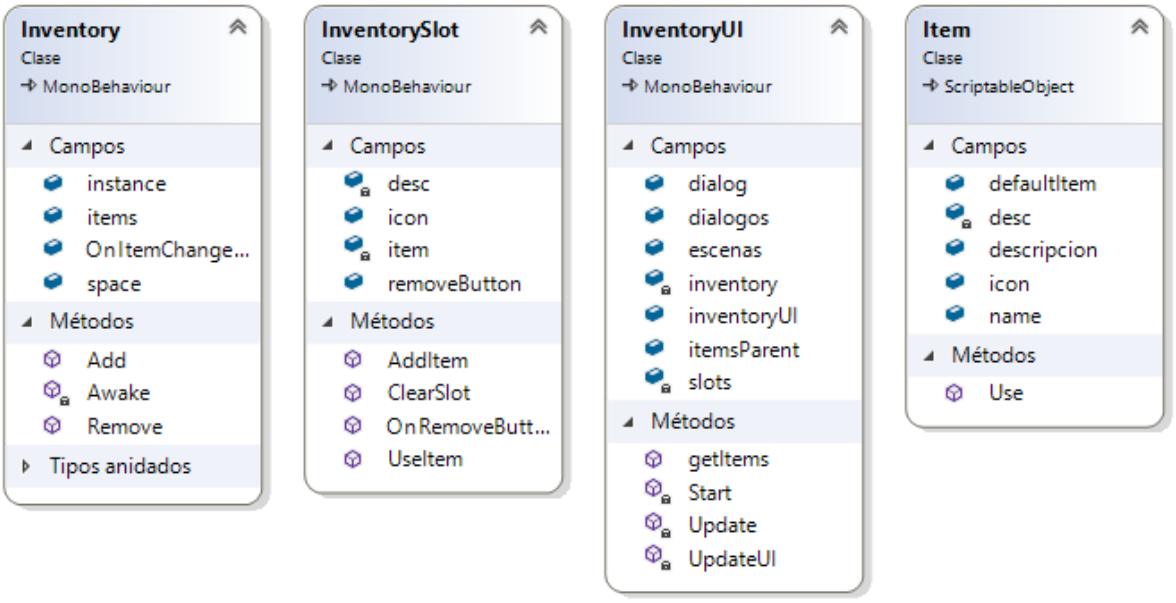

Figura 4 de Diseño: Diagrama de clases Inventario

3.3.4 Diagrama de clases del sistema de eventos

La clase evento se encarga de proporcionar un bloqueo utilizando los objetos que se encuentran en el inventario o de proporcionar un nuevo objeto solo disponible mediante la secuencia de elecciones en un diálogo.

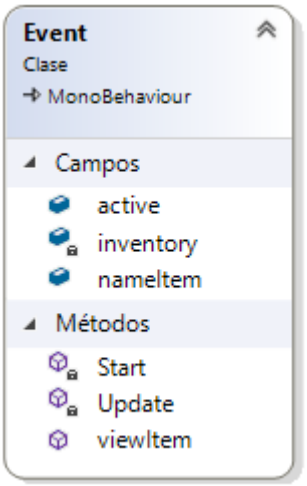

Figura 5 de Diseño: Diagrama de clases Evento

4 Desarrollo

En este apartado se darán los pasos seguidos en la creación de los cuatro sistemas implementados y las decisiones que se tomaron durante su desarrollo en cada uno de ellos.

4.1 Sistema de Diálogo

Para crear los diálogos, se decidió dividir en dos partes los scripts que lo conforman: una parte es el manejador de los diálogos para crear el flujo de las conversaciones y la otra parte es la interfaz para crear diálogos desde Unity.

El manejador de diálogos o Dialog Manager debía de dar forma a los diálogos y el flujo que estos debían de seguir dependiendo de una respuesta. Para cada diálogo se incluyeron tres posibles respuestas en forma de botones, además, se decidió crear un tipo de archivo correspondiente al diálogo, formado a partir de claves llamadas key y con cada key debía de ir un diálogo.

Los textos siguen una estructura para poder crear un flujo entre los diálogos, esta estructura está formada por varias partes:

- Texto del diálogo.
- Los botones con un número para identificarlo, solo hay tres botones y se nombran de la siguiente forma: “button 1”, “button 2” y “button 3”.
- Pequeño texto identificativo para el siguiente diálogo.
- La key del siguiente diálogo o la key “exit” para finalizar el diálogo.
- Tanto el botón, el pequeño texto y la key han de ir entre paréntesis.
- Ejemplos de la estructura:

    Ejemplo de texto con un solo botón. (button 1 Hola key)
    Ejemplo de texto con dos botones. (button 1 Hola key) (button 2 Adios exit)

Esto crea una encadenación de textos y forma una conversación distinta con cada respuesta que seleccione el jugador.

Figura 1 de Desarrollo: Diálogo

En cuanto a la interfaz de creación de diálogos o Dialog System, esta debía ofrecer una simple e intuitiva interfaz para los usuarios, por ello se decidió incluirla como una herramienta en Unity.

La interfaz posee las siguientes funcionalidades:

- Creación de un archivo de diálogo nuevo, únicamente cuando no hay ningún archivo abierto en la interfaz.
- Apertura de un archivo de diálogo existente.
- Se puede añadir o eliminar un nuevo diálogo al archivo.
- Se muestra el número de diálogos del archivo pudiendo desplazarse entre ellos pulsando los botones dedicados a ellos o indicando el número de diálogo al que queremos acceder.
- La clave o key del diálogo.
- Se muestra un ejemplo de la estructura del texto.
- Campo para escribir el texto del diálogo actual.

Figura 2 de Desarrollo: Interfaz de diálogo

4.2 Sistema de Interacciones

La realización de interacciones depende de los elementos con los que se junte pudiendo adaptarse de formas distintas. Debido a esto se decidió crear varias interacciones distintas según lo que se quería conseguir.

Los elementos con interacciones que se decidieron adaptar fueron los siguientes: los NPC que provocan la activación de los diálogos, la recogida de objetos y el cambio de escenarios.

Para poder indicar al usuario que un elemento tiene interacción, se decidió implementar una forma visual para resaltar esta característica. Por ello cuando se posiciona el puntero del ratón sobre un NPC o un objeto se produce un intercambio de punteros. También se produce la aparición de un texto en la parte inferior de la pantalla. Todo esto muestra al usuario la posibilidad de poder interactuar con el elemento seleccionado pulsando el botón izquierdo.

En el script de Zonas Interacción podemos seleccionar entre los elementos de objetos y diálogos en las variables públicas que tiene.

Para poder añadir a una interacción un cambio de escena, debemos de usar el script Cambiar Escenas, esto se hizo así ya que la zona de interacción y el cambio de escena difieren entre sí a la hora de bloquear y desbloquear los GameObject de las escenas.

Para provocar el bloqueo de escenas mediante eventos se ha de crear un script de tipo Event y añadirlo en una de las variables de Cambiar Escenas.

Todas las interacciones de los scripts funcionan sobre todo con los siguientes métodos:

- **Update:** con esta función se comprobarán todos los cambios que se produzcan como pueda ser el boqueo o desbloqueo de escenas o interacciones.

- **OnMouseExit y OnMouseOver:** con estas dos funciones se comprobará si está o no el puntero del ratón posicionado sobre una interacción.

Figura 3 de Desarrollo: Interacción de cambio de escena

4.3 Sistema de Inventario

La creación del inventario se dividió en el propio inventario a partir de una lista de los objetos, la interfaz del inventario del juego y los propios objetos como elementos con los que se puede interactuar.

El inventario está formado por una lista de objetos y un determinado espacio variable de objetos. Los objetos pueden añadirse a la lista al interactuar con un objeto del escenario y si hay espacio libre en el inventario actual.

En la interfaz del inventario se empleó el uso de huecos o slots para cada objeto y un cuadro de texto para mostrar la descripción de los objetos. Cuando se añade un objeto al inventario, este se añadirá en el primer hueco disponible desde la parte superior izquierda.
Al clicar sobre un objeto se mostrará su descripción en la parte derecha de la pantalla. Además, en la parte superior derecha de cada objeto hay un botón para poder eliminar el objeto del inventario.

Los objetos están formados por un nombre, una descripción y un icono. Todos los objetos dispuestos en las escenas han de ir acompañados por el script Zona Interacción.

Por último, el inventario ha de poder mostrarse de una forma sencilla y para ello se adaptó el botón derecho del ratón para poder abrir y cerrar el inventario del jugador.

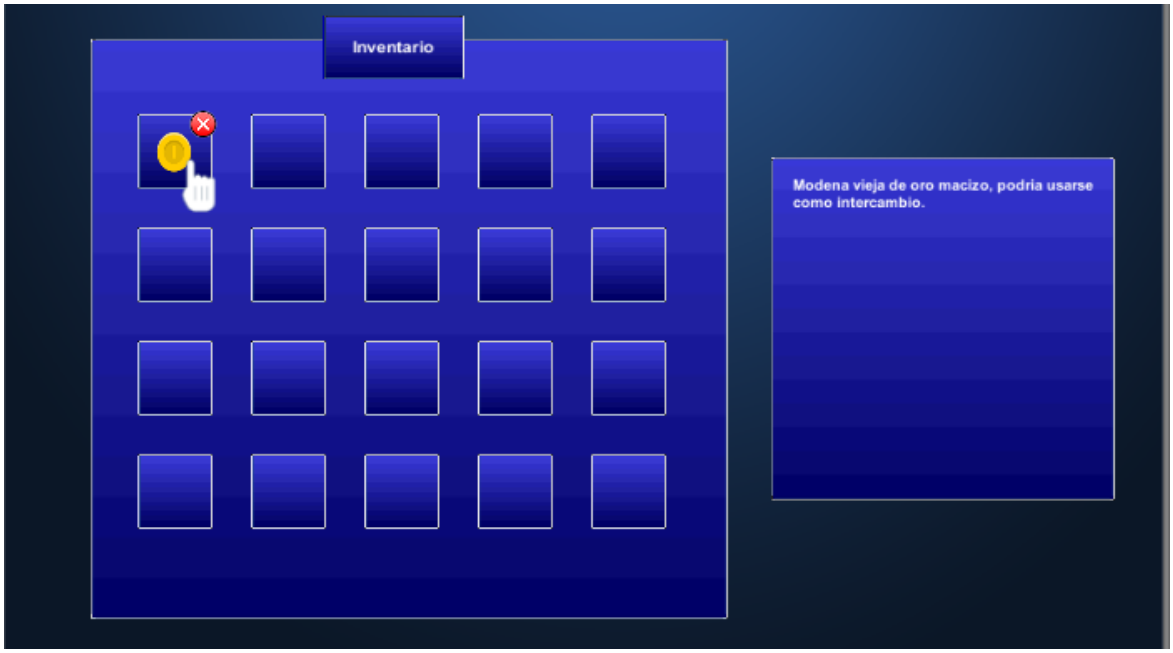

Figura 4 de Desarrollo: Inventario

4.4 Sistema de Eventos

Los eventos se han implementado a partir de los demás sistemas siendo más un añadido y un complemento que un sistema propio en sí, es decir, el tipo de evento que se quería obtener dependía de los otros sistemas.

En las interacciones se ha añadido la posibilidad de habilitar un evento de bloqueo, complementado a partir de unos determinados objetos localizados en el inventario.

En los diálogos e inventario se ha desarrollado la posibilidad de obtener un objeto específico según las decisiones que se han ido tomando en el transcurso de un diálogo. Estos objetos pueden desbloquear un evento de una interacción.

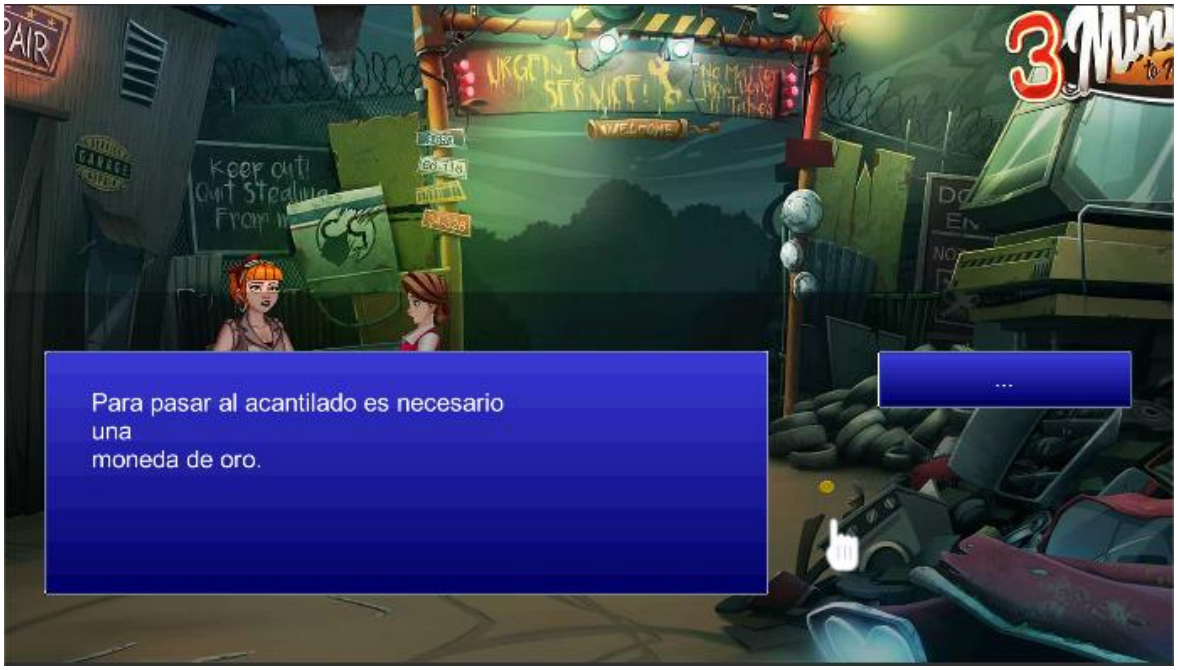

Figura 5 de Desarrollo: Evento para cambio de escena

5 Integración, pruebas y resultados

Para comprobar el correcto funcionamiento del Asset y de los scripts en este apartado se realizarán las pruebas de forma conjunta formando una pequeña demo jugable. Con ello se consigue ver un prototipo de aventura gráfica generado con los scripts implementados.

5.1 Implementación

Se ha implementado una demo donde se pretende probar la funcionalidad en conjunto e individualmente de los cuatro sistemas mencionados anteriormente.

La demo está formada por los siguientes GameObjects:

- La creación de los dos assets implementados, los diálogos y los objetos, con sus correspondientes descripciones, atributos y scripts.
- Implementación del inventario con su correspondiente canvas, un número determinado de objetos máximos y la capacidad para añadir o eliminar objetos.
- El canvas del inventario está formado por un número de huecos, los cuales han de ser los mismos que el tamaño máximo del inventario. Cada objeto en el inventario puede pulsarse para obtener una descripción de este y un botón para eliminarlo.
- En la interfaz del diálogo se crean nuevos diálogos con sus correspondientes claves y textos, además de añadir o eliminar nuevos textos.
- En el canvas del diálogo se implementan todos los diálogos existentes con sus correspondientes claves, textos y botones.
- La agrupación de todas las implementaciones de las escenas disponibles, donde cada una de ellas se añaden los elementos con los que poder interactuar como bien puedan ser los objetos, diálogos o cambios de escenas.
- Los eventos se implementan en cada uno de los elementos de interacción bloqueando el paso hacia nuevas escenas a cambio de objetos y conversaciones con recompensas de nuevos objetos.

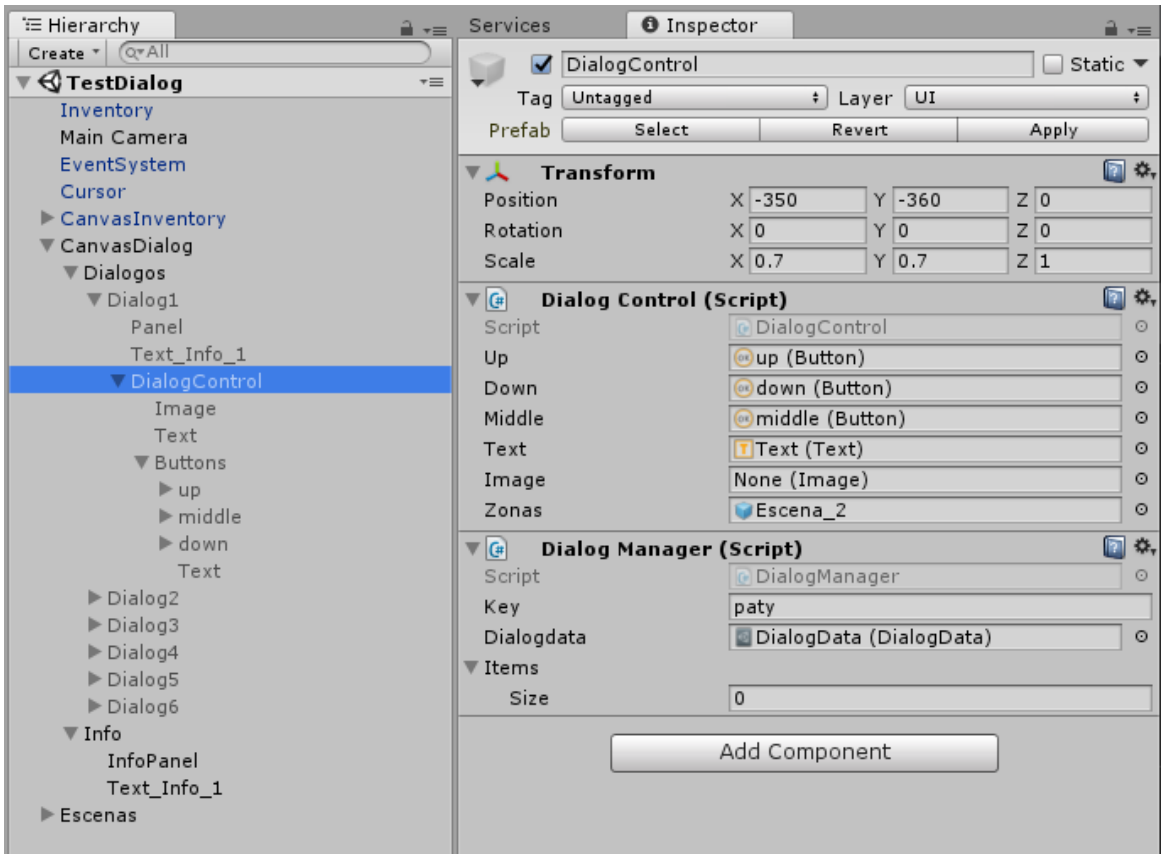

Figura 1 de Implementación: GameObject DialgogControl

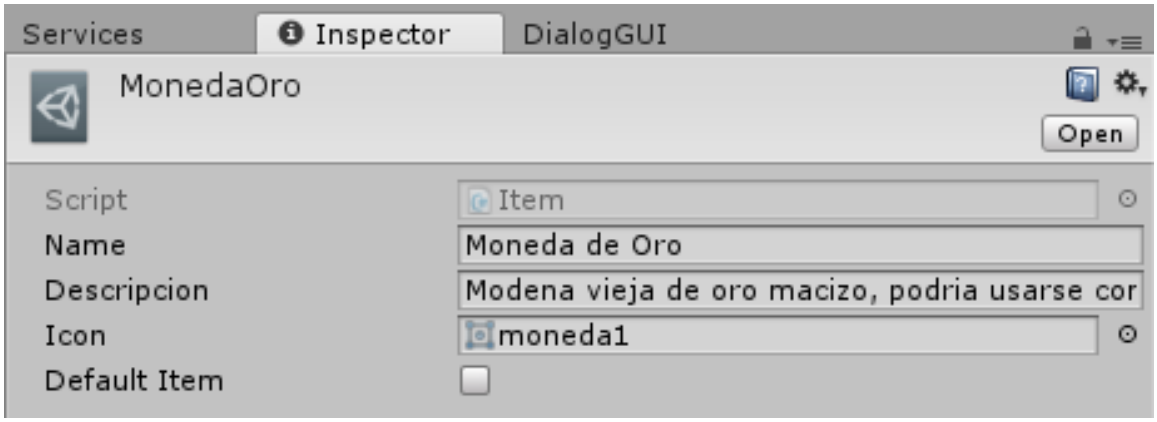

Figura 2 de Implementación: GameObject de Objeto

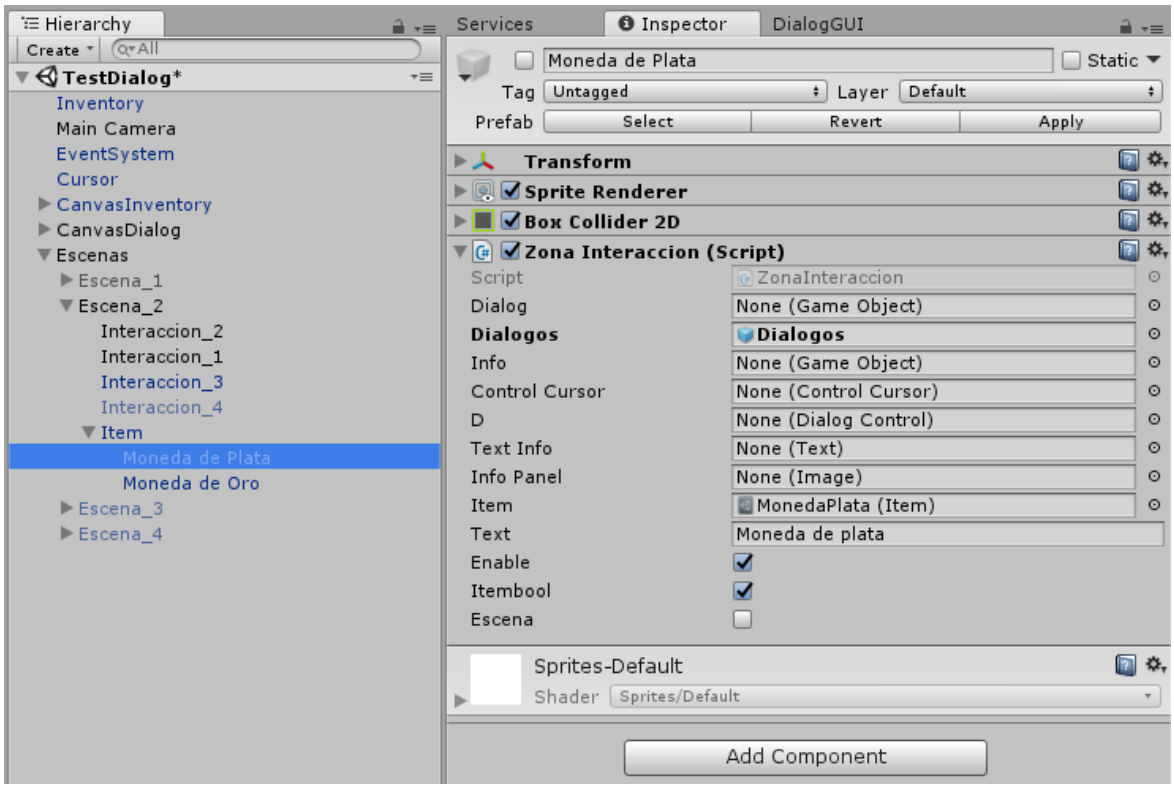

Figura 3 de Implementación: GameObject de Interacción de objeto

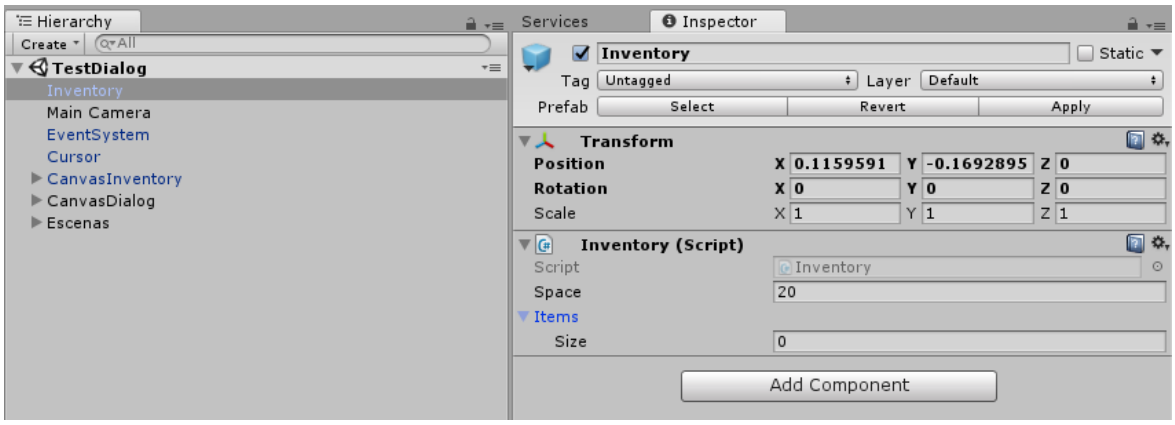

Figura 4 de Implementación: GameObject de Inventory

5.2 Pruebas y resultados

Para comprobar y asegurar el cumplimiento de los objetivos establecidos se realizaron varias pruebas del software, llevando a cabo tanto pruebas individuales como colectivas de los sistemas creados.

Se emplearon dos tipos de pruebas, las unitarias y las de integración:

Las pruebas unitarias se centraban en asegurar la ejecución sin errores del código y de comprobar que cada sistema realice sus funciones correctamente.

Las pruebas de integración como su propio nombre indica se enfocan en integrar cada sistema para que pueda funcionar en su conjunto como si fuera un único sistema. La mayoría de las pruebas comprobaban la interacción de las funciones de los cuatro sistemas a la vez.

Como hay cuatro sistemas se hablará de cada una de las pruebas realizadas en cada uno de ellos.

Sistema de diálogos:

Pruebas unitarias:

- Creación de un nuevo <dialogo.asset>, con sus campos vacíos para poder rellenarse y con la apertura de este diálogo en la interfaz. Figura AnexoC-32, Figura AnexoC-33, Figura AnexoC-34 y Figura AnexoC-37.
- En la interfaz se probaron la apertura de otros diálogos, añadir y eliminar diálogos. Figura AnexoC-34, Figura AnexoC-35 y Figura AnexoC-36.
- La interacción correcta del diálogo actual con los botones enlazando estos botones al pulsarse con los siguientes diálogos. Figura AnexoC-38.
- Realización de pruebas con introducción de textos en un formato incorrecto.

Pruebas de integración:

- Prueba correcta de bloqueo de todos los diálogos y las interacciones excepto el que se encuentra activo en ese momento.
- Prueba correcta de bloqueo del inventario.

No se realizaron pruebas referentes a la sintaxis que han de seguir los diálogos para el funcionamiento de estos y de los botones, ya que esta sintaxis ha de seguir una estructura concreta para que funcionen las transiciones de los diálogos y las interacciones con los botones.

En las pruebas de los diálogos se obtuvieron resultados satisfactorios y se pueden apreciar con imágenes en el apartado C de los Anexos.

Sistema de interacciones:

Pruebas unitarias:

- Prueba correcta en la mayoría de los casos sobre el cambio de puntero del ratón cuando este se posiciona sobre una interacción y el reinicio del puntero cuando ya no se encuentra sobre la interacción. Figura AnexoC-39 y Figura AnexoC-41.
- Prueba correcta sobre la visualización de información cuando se posiciona el ratón sobre una interacción. Figura AnexoC-39, Figura AnexoC-38, Figura AnexoC-40 y Figura AnexoC-42.
- Al clicar sobre una interacción, esta responde correctamente adecuándose al tipo de interacción que se le ha dado.

Pruebas de integración:

- Prueba correcta de bloqueo de todas las interacciones hasta que termine la acción que se ha activado.
- En el cambio de escena se ha comprobado que se activen todas las interacciones de la escena siguiente y que se desactiven todas las interacciones de la escena anterior.
- Al interactuar con un objeto, este desaparece y se añade al inventario correctamente. Figura AnexoC-40 y Figura AnexoC-41.

En los resultados de las interacciones se obtuvieron resultados satisfactorios, excepto en algunos momentos del cambio del puntero del ratón en los cuales tardaba unos segundos en responder. Esto se producía cuando se realizaban varias interacciones seguidas.

Algunos de los resultados pueden observarse en el apartado C de los Anexos.

Sistema de inventario:

Pruebas unitarias:

- Si se clica el botón derecho del ratón se abre y se cierra el inventario en el juego correctamente. Figura AnexoC-45.
- Al pulsar un objeto del inventario aparece correctamente una descripción del objeto en la parte derecha de la pantalla. Figura AnexoC-46 y Figura AnexoC-47.
- Si se ha pulsado un objeto del inventario para ver la descripción y después se pulsa otro objeto distinto al anterior, se cambia la descripción a la del último objeto pulsado correctamente.
- Si se ha pulsado un objeto del inventario, para ver la descripción y después se ha pulsado el botón <X> de la esquina superior derecha de ese mismo objeto se borra el objeto y la descripción desaparece. Figura AnexoC-47 y Figura AnexoC-45.
- Se añade correctamente el objeto con el que se ha interactuado al inventario si hay espacio disponible. Figura AnexoC-47.

En los resultados del inventario se obtuvieron resultados satisfactorios, algunos de los resultados pueden observarse en el apartado C de los Anexos.

Sistema de eventos:

Pruebas unitarias:

- Bloqueo de la interacción propia hasta que se cumplan ciertos requisitos adecuadamente. Figura AnexoC-51.
- Si se bloquea con ciertos objetos, se comprueba que esos objetos se encuentran en el inventario correctamente, desbloqueando o no la interacción. Figura AnexoC-47 y Figura AnexoC-51.
- Si se desbloquea el evento, este desaparece correctamente. Figura AnexoC-47, Figura AnexoC-51 y Figura AnexoC-42.

Pruebas de integración:

- Si el evento desbloquea un objeto, este se añade correctamente al inventario si hay espacio disponible. Figura AnexoC-48, Figura AnexoC-49 y Figura AnexoC-50.

En las pruebas relacionadas con los eventos se obtuvieron resultados correctos y se pueden observar con imágenes en el apartado C de los Anexos.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Antes de realizar este trabajo de fin de grado cursé la asignatura de videojuegos con la cual aprendí a usar el motor gráfico de Unity y a familiarizarme con la interfaz, además de eso también aprendí lo básico para entender y crear scripts sencillos en el lenguaje de programación C# para crear elementos y acciones en Unity. Todo lo que aprendí durante los meses que cursé esta asignatura me sirvió para realizar este trabajo fin de grado.

El uso de un motor gráfico como el de Unity nos da la capacidad de crear un videojuego desde cero o usando las herramientas ya creadas por otros usuarios. En el caso de este proyecto como había que crear estas herramientas he tenido algunas dificultades, ya que en un principio la creación de los scripts se me complicó por la falta de conocimiento de ciertas funciones usadas en Unity. Sin embargo, tras investigar en el propio manual y guías sobre scripting que tiene Unity aprendí los conocimientos necesarios para manejar las funciones que usaría en C#.

A la hora de decidir las herramientas que se iban a desarrollar, en un principio lo tenía claro pero no sabía hasta qué punto iba a ser capaz de desarrollarlas, esa incertidumbre me dejó algunos días pensativo y sin saber qué hacer. Al final lo que hice fue hacerme un esquema de las cosas que tenía que hacer, y comencé agrupando las herramientas en sistemas según lo que debían hacer. Cabe mencionar que realicé una búsqueda de herramientas similares para comprobar la capacidad que tenía y saber hasta qué punto podía desarrollarlas.

Durante la realización de este trabajo fin de grado, he de decir que he podido cumplir con la mayoría de los requisitos que me había propuesto desde el inicio de este. Entre estos se encontraban la capacidad de aumentar los conocimientos que tenía sobre el manejo del motor de videojuegos Unity y el lenguaje de programación C#, establecer los sistemas o herramientas que iban a dar forma al proyecto y seguir un orden a la hora de desarrollar cada uno.

6.2 Trabajo futuro

En este proyecto se han tratado las mecánicas principales de los juegos de aventuras gráficas, sin embargo, todas estas pueden mejorarse con añadidos o incluyendo otras mecánicas que las complementen.

En este apartado se alistarán posibles implementaciones para hacer en el futuro.

- Actualmente, la mayoría de los juegos apuestan por plataformas 3D. Es por ello por lo que se podrían adaptar las herramientas a las tres dimensiones dando así una mayor capacidad de elección a los desarrolladores.

- Los eventos que hemos creado se pueden ampliar de forma considerable dando más opciones. Algunas de las posibles implementaciones que se podrían hacer serían dar acciones según un tipo de objeto o poder bloquear ciertos diálogos según los objetos que tengas o decisiones que se tomaron en los diálogos anteriores.

- Mantenimiento del proyecto software, delegando al futuro la corrección de errores que se hayan ido reportado.

- Desarrollar una mayor interacción con los objetos obtenidos, lo que puede realizar intercambios o puzles.

- Dar mayor viveza al juego y sus escenarios con la posibilidad de que los personajes tengan movilidad.

- Desarrollo de nuevas herramientas dedicadas a guardar y cargar la partida, menú de opciones y de audio.

- Añadir más mecánicas propias de las aventuras gráficas, como pueden ser la toma de decisiones con tiempo, realizar puzles o minijuegos, etc.

Referencias

[1] “Pong”, Julio 2019, https://es.wikipedia.org/wiki/Pong#/media/Archivo:Pong.png
[2] “2D Physics”, Julio 2019, https://learn.unity.com/tutorial/2d-physics
[3] “Physics in Unity 5.0”, Julio 2019, https://docs.unity3d.com/Manual/UpgradeGuide5-Physics.html
[4] “Desarrollo de juegos con Unity 3D ¿Cómo funciona esta herramienta?”, Julio 2019, https://www.yeeply.com/blog/desarrollo-de-juegos-con-unity-3d/
[5] “Unity 2019: Performance by Default, gráficos de alta fidelidad en tiempo real y herramientas para artistas”, Julio 2019, https://unity3d.com/es/unity
[6] “Unity”, Julio 2019, http://www.pulso.uniovi.es/wiki/index.php/Unity
[7] “Unity (motor de videojuego)”, Julio 2019, https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)
[8] “Unity, el motor de desarrollo capaz de partir la historia de los videojuegos en dos”, Dani Candil, Julio 2019, https://www.vidaextra.com/industria/unity-el-motor-de-desarrollo-capaz-de-partir-la-historia-de-los-videojuegos-en-dos
[9] “The Secret of Monkey Island”, Julio 2019, https://www.3djuegos.com/juegos/analisis/lectores/4221/0/the-secret-of-monkey-island/
[10] “Historia y cultura popular en Monkey Island”, Julio 2019, https://www.presura.es/blog/2017/01/30/historia-cultura-popular-monkey-island/
[11] “The Secret of Monkey Island”, Julio 2019, https://es.wikipedia.org/wiki/The_Secret_of_Monkey_Island
[12] “Aventura conversacional”, Agosto 2019, https://es.wikipedia.org/wiki/Aventura_conversacional
[13] “HISTORIA DE LAS AVENTURAS GRÁFICAS I”, Iván Diez, Agosto 2019, https://juegosadn.eleconomista.es/articulo-historia-de-las-aventuras-graficas-i-ar-604/
[14] “HISTORIA DE LAS AVENTURAS GRÁFICAS II”, Iván Diez, Agosto 2019, https://juegosadn.eleconomista.es/articulo-historia-de-las-aventuras-graficas-ii-ar-674/
[15] “HISTORIA DE LAS AVENTURAS GRÁFICAS III”, Iván Diez, Agosto 2019, https://juegosadn.eleconomista.es/articulo-historia-de-las-aventuras-graficas-iii-ar-523/
[16] “Historia de los videojuegos”, Agosto 2019, https://www.fib.upc.edu/retro-informatica/historia/videojocs.html
[17] “The sspriters ressource”, Agosto 2019, https://www.spriters-resource.com/
[18] “Singleton”, Septiembre 2019, https://es.wikipedia.org/wiki/Singleton
[19] “Abstract Factory”, Septiembre 2019, https://es.wikipedia.org/wiki/Abstract_Factory
[20] “Unity 2D”, Septiembre 2019, https://docs.unity3d.com/es/530/Manual/Unity2D.html
[21] “SCUMM”, Septiembre 2019, https://es.wikipedia.org/wiki/SCUMM#Véase_también
[22] “Sprites”, Septiembre 2019, https://docs.unity3d.com/es/530/Manual/Sprites.html
[23] “Bases de Unity”, Septiembre 2019, https://docs.unity3d.com/es/530/Manual/UnityBasics.html
[24] “Creación de Juego”, Septiembre 2019, https://docs.unity3d.com/es/530/Manual/CreatingGameplay.html
[25] “Scripting”, Septiembre 2019, https://docs.unity3d.com/es/530/Manual/ScriptingSection.html

[26] "Visionaire studio", Octubre 2019, https://www.visionaire-studio.net
[27] "Visionaire studio forum", Octubre 2019, https://www.visionaire-studio.net/forum/
[28] "Adventure game studio forums", Octubre 2019, https://www.adventuregamestudio.co.uk/forums/index.php?topic=53617.0
[29] "Visionaire studio wiki", Octubre 2019, https://www.visionaire-studio.net/cms/wiki-2023.html
[30] "Visionaire studio wikipedia", Octubre 2019, https://de.wikipedia.org/wiki/Visionaire
[31] "SCUMM wikipedia", Octubre 2019, https://en.wikipedia.org/wiki/SCUMM
[32] "SCUMMVM", Octubre 2019, https://www.scummvm.org
[33] "SCUMM wikipedia", Octubre 2019, https://en.wikipedia.org/wiki/ScummVM
[34] "SCUMM wikipedia Gob", Octubre 2019, https://wiki.scummvm.org/index.php?title=Gob
[35] "SCUMM wikipedia Engine", Octubre 2019, https://wiki.scummvm.org/index.php/Engines
[36] "Bidimensional", Diciembre 2019, https://es.wikipedia.org/wiki/Bidimensional
[37] "Tridimensional", Diciembre 2019, https://es.wikipedia.org/wiki/Tridimensional
[38] "SCUMM wikipedia AGI", Octubre 2019, https://wiki.scummvm.org/index.php?title=AGI
[39] "Wintermute Engine forum", Octubre 2019, http://forum.dead-code.org/index.php?topic=4771.0
[40] "Wintermute Engine(CREA AVENTURAS GRAFICAS)" Octubre 2019, http://chemyrok.over-blog.es/2014/11/wintermute-engine-crea-aventuras-graficas.html
[41] "Wintermute Engine wikipedia", Octubre 2019, https://en.wikipedia.org/wiki/Wintermute_Engine
[42] "Wintermute Engine home", Octubre 2019, http://dead-code.org/home/
[43] "Adventure game studio", Noviembre 2019, https://www.adventuregamestudio.co.uk/site/ags/
[44] "Adventure game studio wikipedia", Noviembre 2019, https://en.wikipedia.org/wiki/Adventure_Game_Studio
[45] "2.5D", Diciembre 2019, https://es.wikipedia.org/wiki/2.5D

Glosario

API	Application Programming Interface
Asset	Denominación que se le da a las librerías para Unity.
Unity	Programa con el que se puede desarrollar un juego.
2D	Juegos desarrollados unicamente en dos dimensiones.
2.5D	Juegos desarrollados con perspectiva 3/4 y pseudo 3D.
3D	Juegos desarrollados unicamente en tres dimensiones.
Multiplataforma	Juego que puede funcionar en diversas plataformas.
Aventura gráfica	Género de videojuego.
Motor gráfico	Herramientas para desarrollar un videojuego
C Sharp	Lenguaje de programación.
NPC’s	Personajes interactuables con el jugador.
GameObjects	Clase del motor Unity.
Canvas	Área donde todos los elementos UI deben estar.
Singleton	Patrón de diseño.
Slot	Huecos o espacios para almacenaje.

Anexos

A Manual de instalación

Instalación o importación del Asset:

En Unity el proceso de instalación se produce a través de los Assets o también llamadas librerías y para realizar la instalación de estas librerías se deberá seguir las siguientes instrucciones:

1. Una vez descargado el asset, en este caso se llama "GraphicAdventureGenerator.unitypackage", y se ha creado un nuevo proyecto de unity, se puede instalar de las siguientes formas:

   Primera forma de hacerlo:

   Solo se tendría que arrastrar el Asset del proyecto a la zona de Assets del nuevo proyecto.

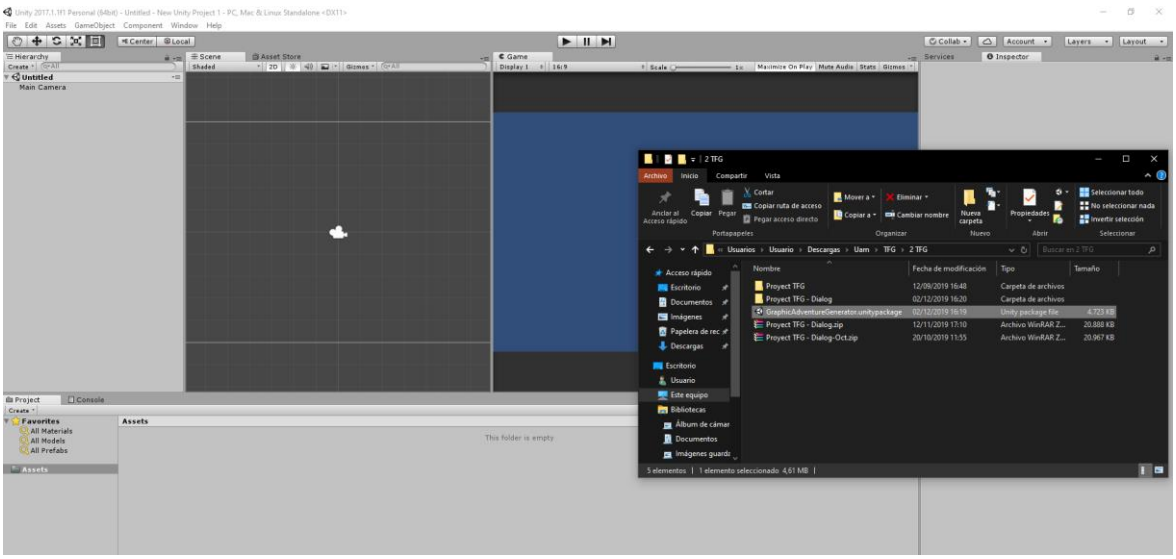

Figura AnexoA-1: Instalacción 1

Segunda forma de hacerlo:

Se pulsará la pestaña de "Assets" ubicada en la parte superior izquierda del programa, seguidamente se buscará "Import Package" y por último se seleccionará la opción llamada "Custom Package…".

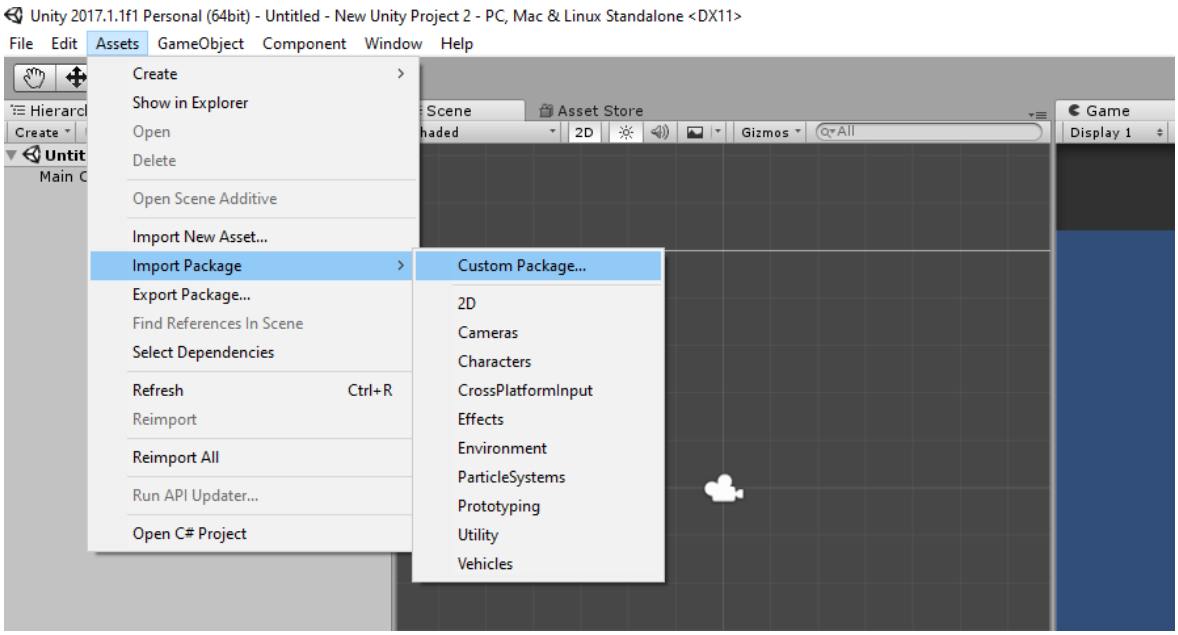

Figura AnexoA-2: Instalacción 2

Seguidamente se buscará la ubicación del Assets y se tendrá que abrir.

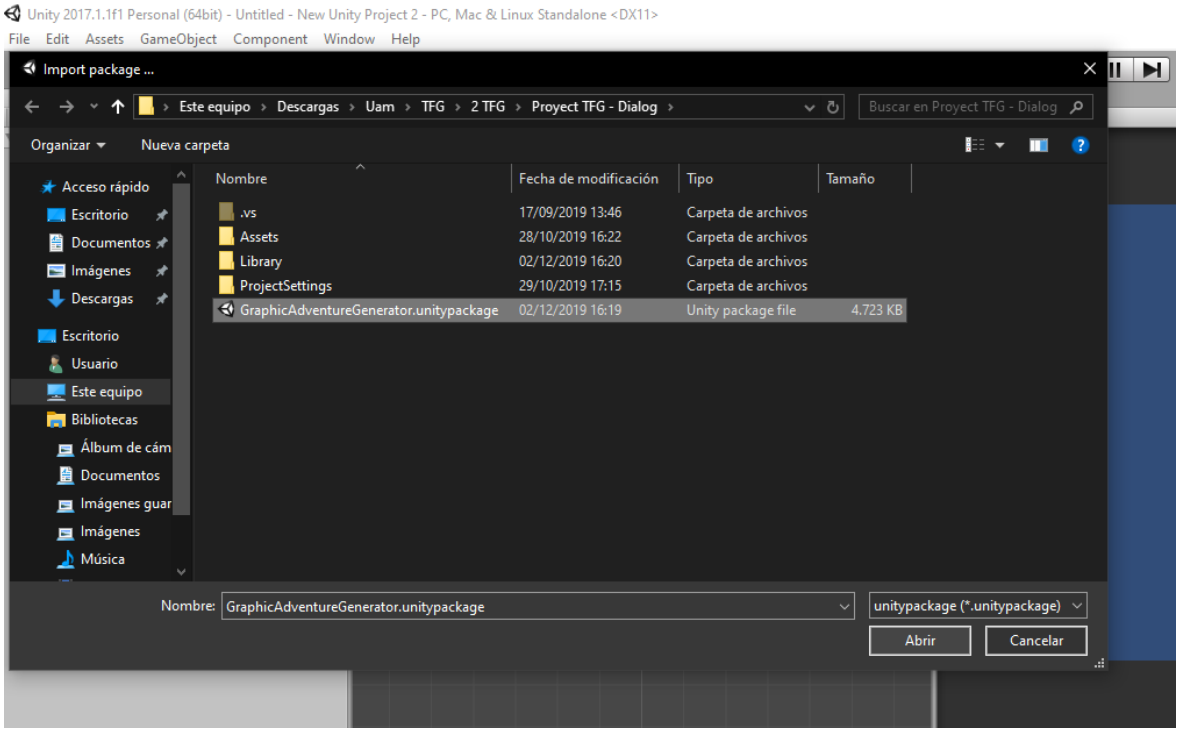

Figura AnexoA-3: Instalacción 3

2. Si bien se han seguido cualquiera de las dos formas mencionadas arriba, aparecerá la siguiente ventana, donde se seleccionarán de forma individual los Assets que queremos importar. En este caso se seleccionarán todos.

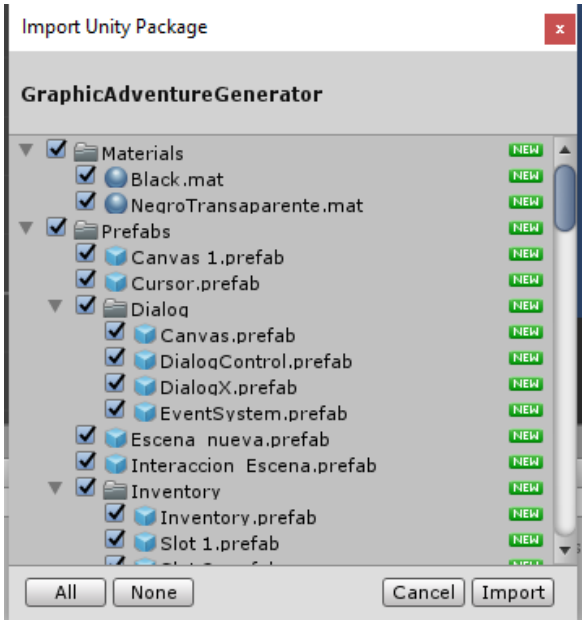

Figura AnexoA-4: Instalacción 4

3. Una vez se han importado los Assets necesarios, estos estarán ubicados en la pestaña de Proyect del programa en forma de carpetas y archivos.

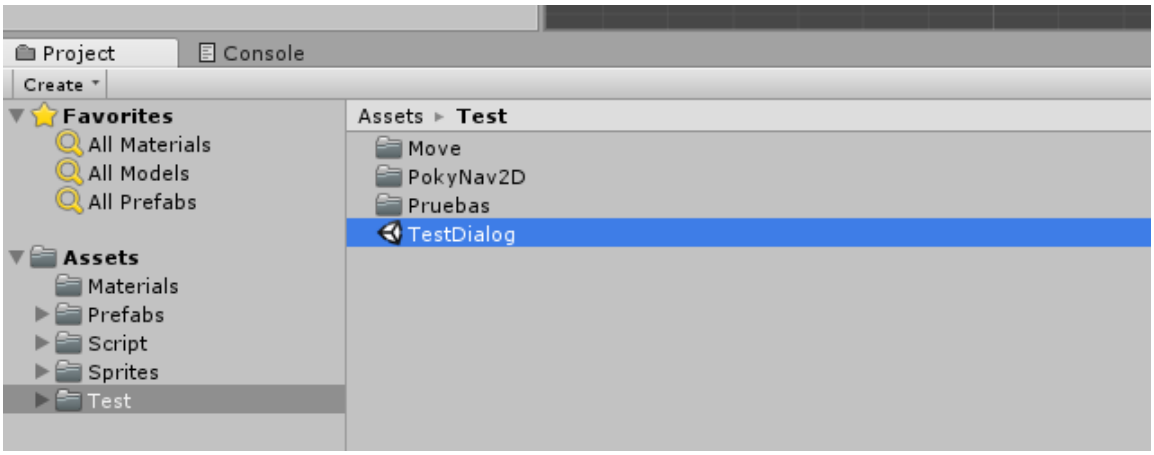

Figura AnexoA-5: Instalacción 5

4. Y, por último, para probar la demo, hay que ir a la carpeta de Test y abrir la escena llamada “TestDialog”.

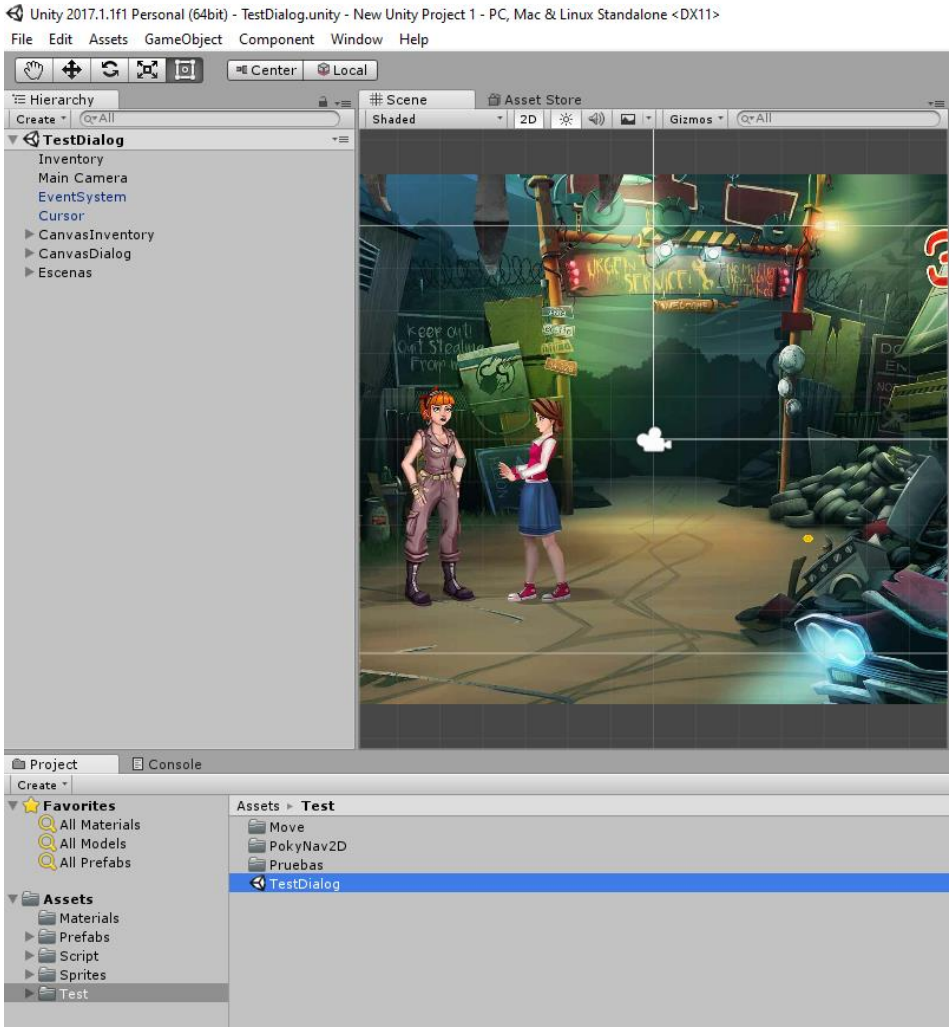

Figura AnexoA-6: Instalacción 6

B Manual del Asset

Estructura de las carpetas del Asset:

- **Carpeta Global:** esta es la estructura global del Asset con todas las carpetas y las subcarpetas que posee, más adelante se hablará de cada una de ellas.

  Las carpetas que cabe destacar entre las demás son las correspondientes a la de los scripts, los prefabs y la de test.

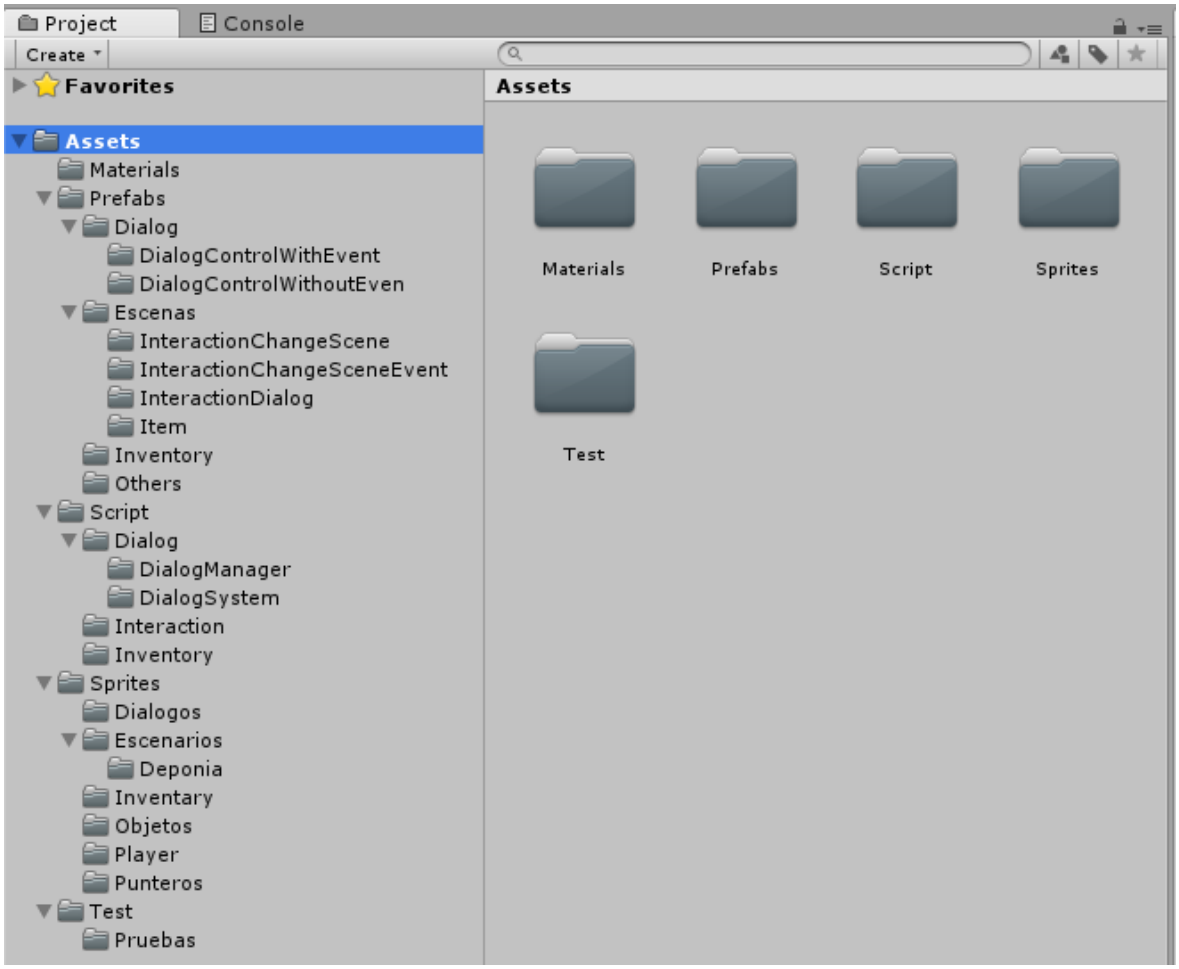

Figura AnexoB-7: Carpeta Global

- **Carpeta de los materiales:** estos son los materiales usados.

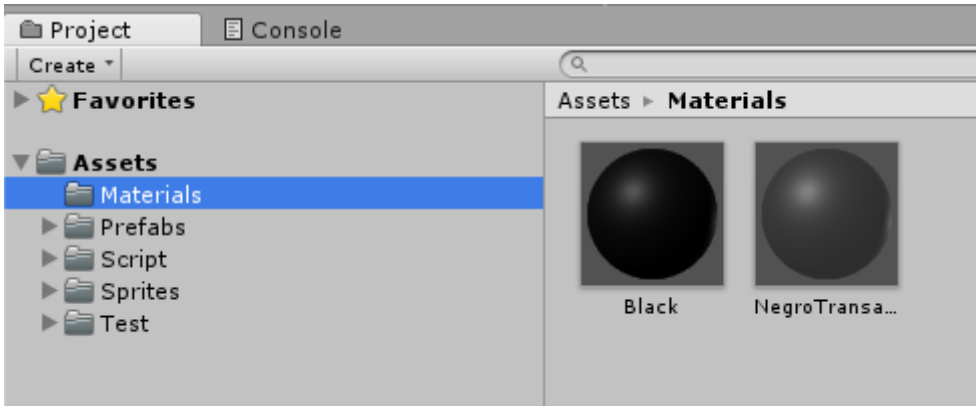

Figura AnexoB-8: Carpeta de Materials

- **Carpeta de los prefabs:** aquí se encontrarán los prefabs correspondientes a la mayoría de los objetos que se han desarrollado.

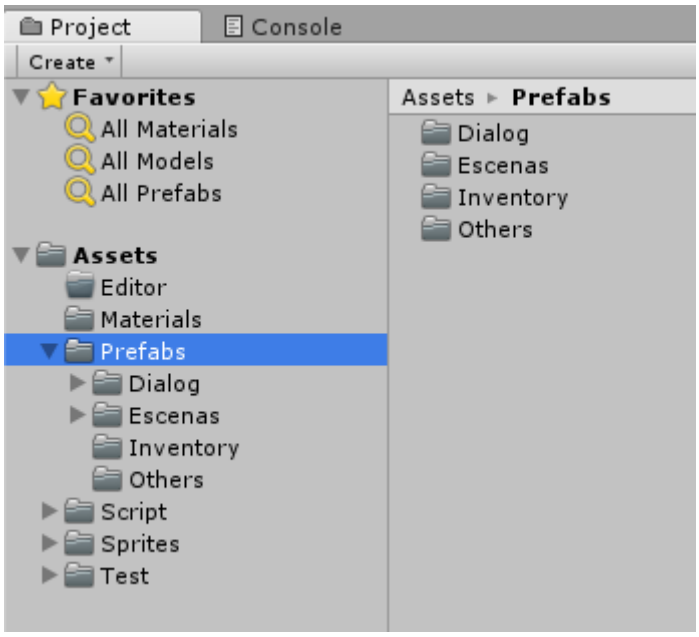

Figura AnexoB-9: Carpeta de Prefabs

Estos son los prefabs del sistema de diálogos tanto individualmente como en conjuntos pudiendo añadir un único diálogo (DialogX) o toda la implementación (CanvasDialog) en la que se incluyen varios diálogos de ejemplo.

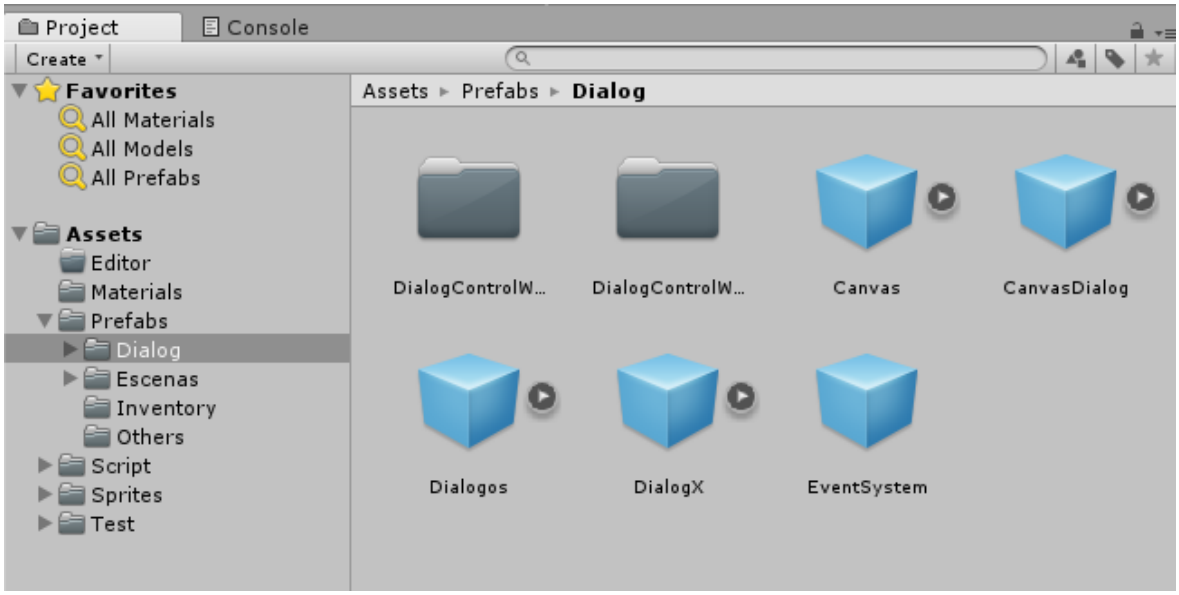

Figura AnexoB-10: Carpeta de Prefabs de Dialog

Estos son los prefabs del sistema de inventario, los más importantes son InventoryParameter y CanvasInventory, con ellos dos se establecerá todo el sistema de inventario.

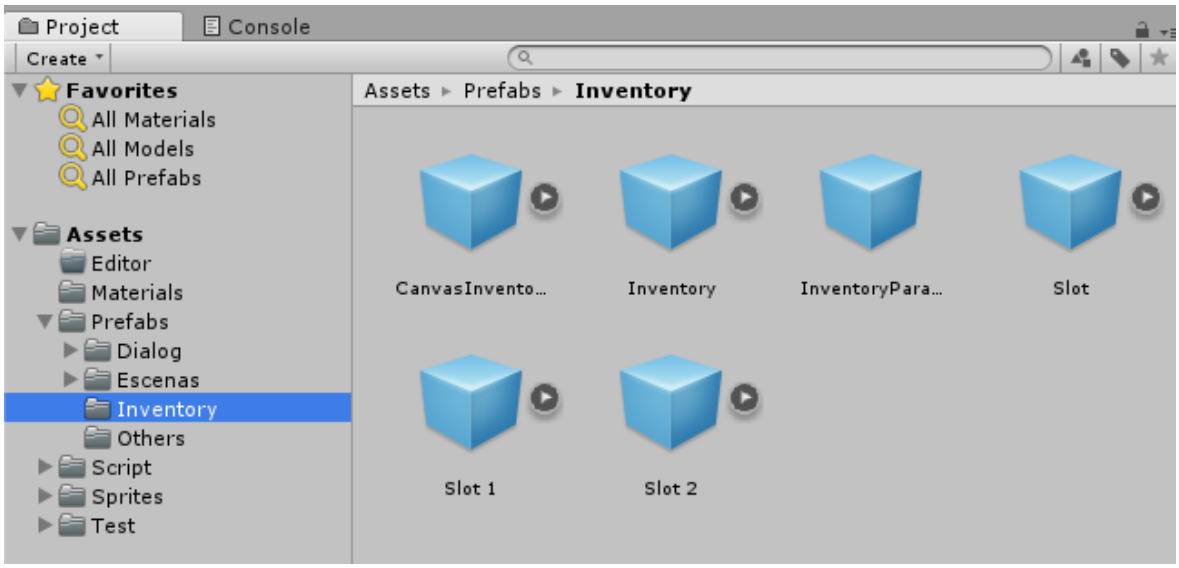

Figura AnexoB-11: Carpeta de Prefabs de Inventory

Estos son los prefabs del sistema de eventos, interacciones y objetos. Aquí se podrán encontrar diferentes tipos de escenas, objetos e interacciones con o sin eventos.

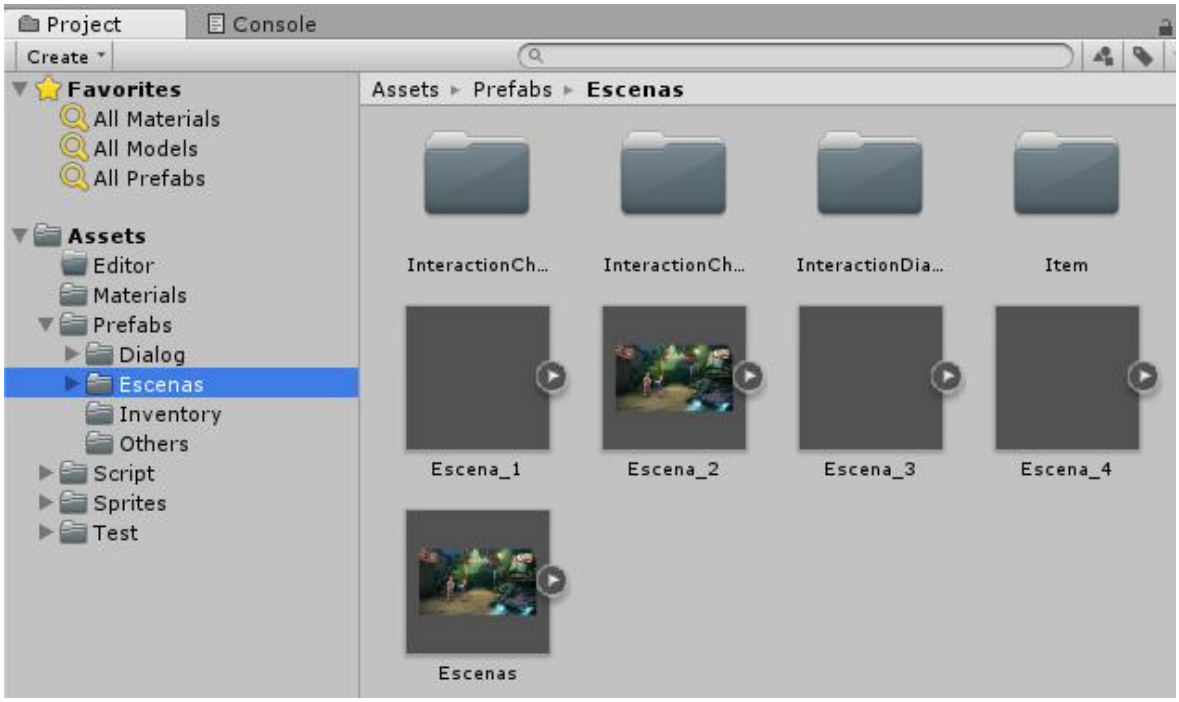

Figura AnexoB-12: Carpeta de Prefabs de Escenas

- **Carpeta de los scripts:** estos son los scripts usados.

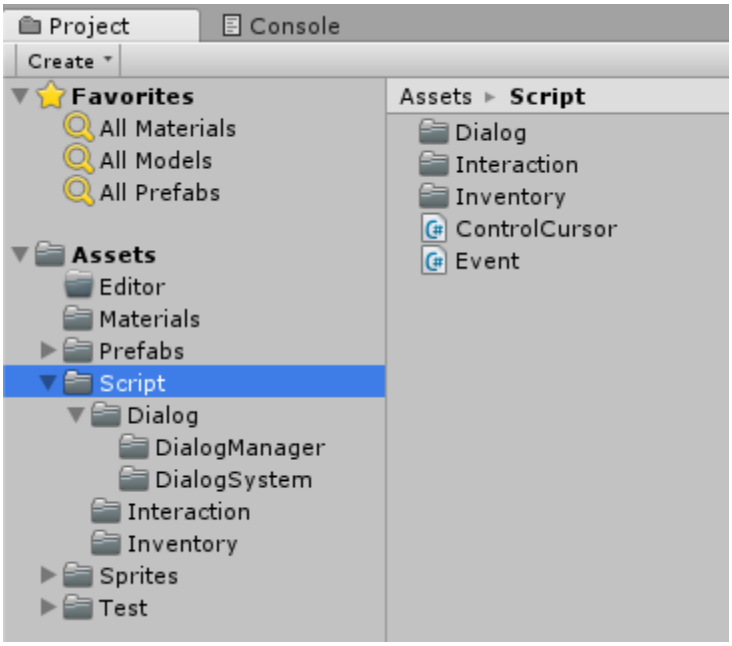

Figura AnexoB-13: Carpeta de Scripts

Estos son los scripts correspondientes al sistema de diálogos y los archivos de datos de los diálogos.

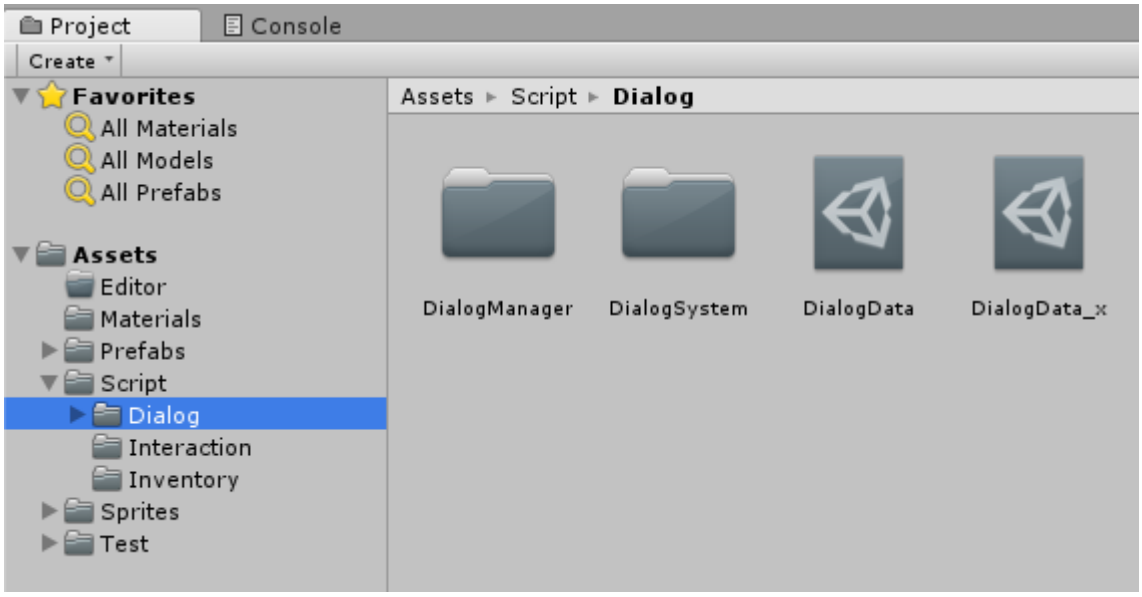

Figura AnexoB-14: Carpeta de Scripts de Dialog

Estos son los scripts correspondientes al sistema de interacción.

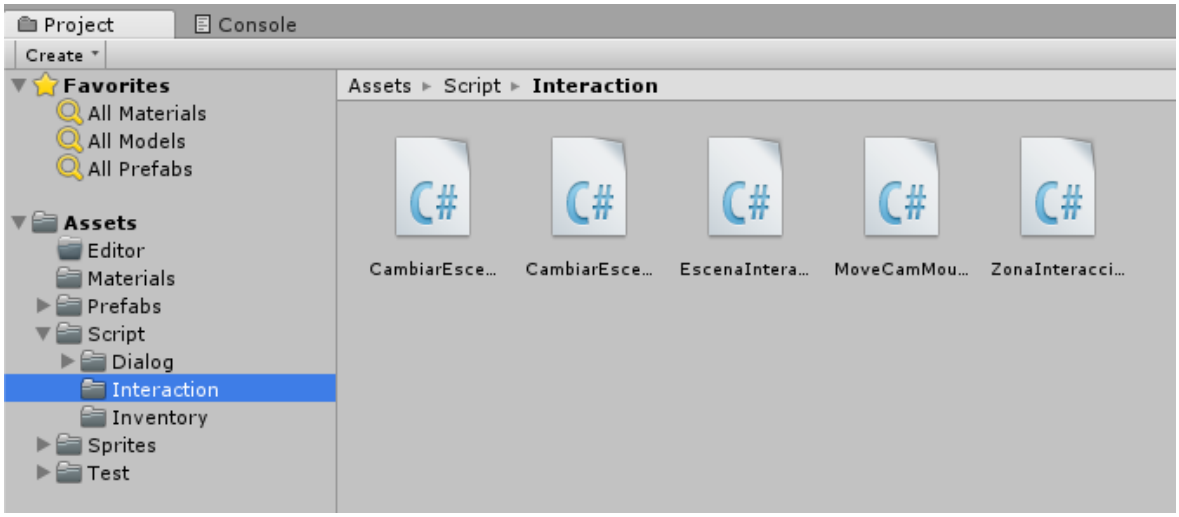

Figura AnexoB-15: Carpeta de Scripts de Interaction

Estos son los scripts correspondientes al sistema de inventario y algunos archivos de datos.

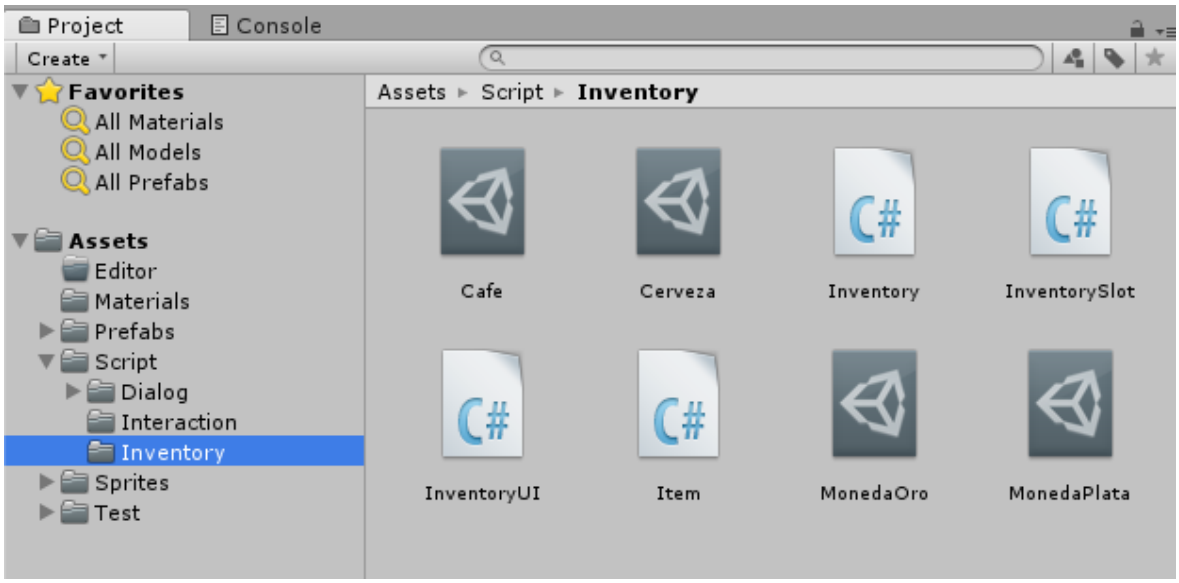

Figura AnexoB-16: Carpeta de Scripts de Inventory

- **Carpeta de los Sprites:** estos son los sprites usados.

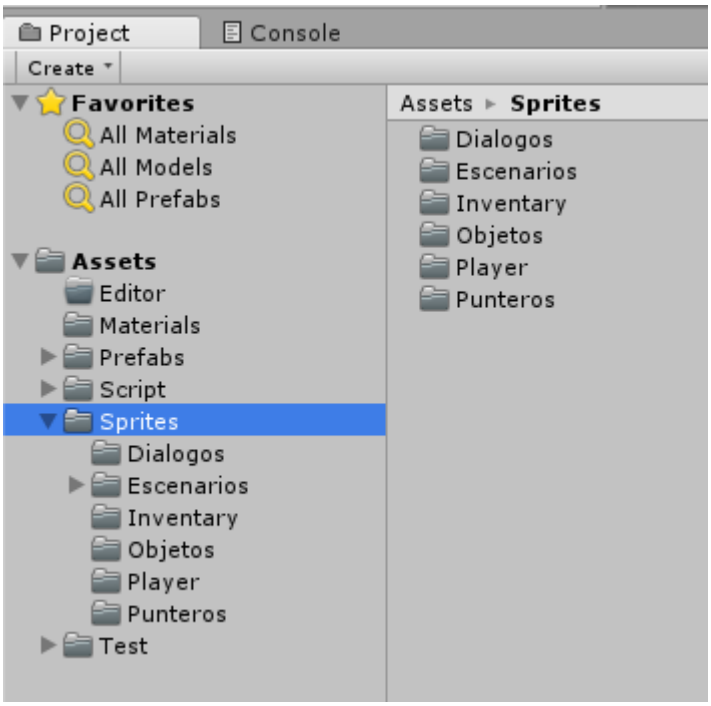

Figura AnexoB-17: Carpeta de Sprites

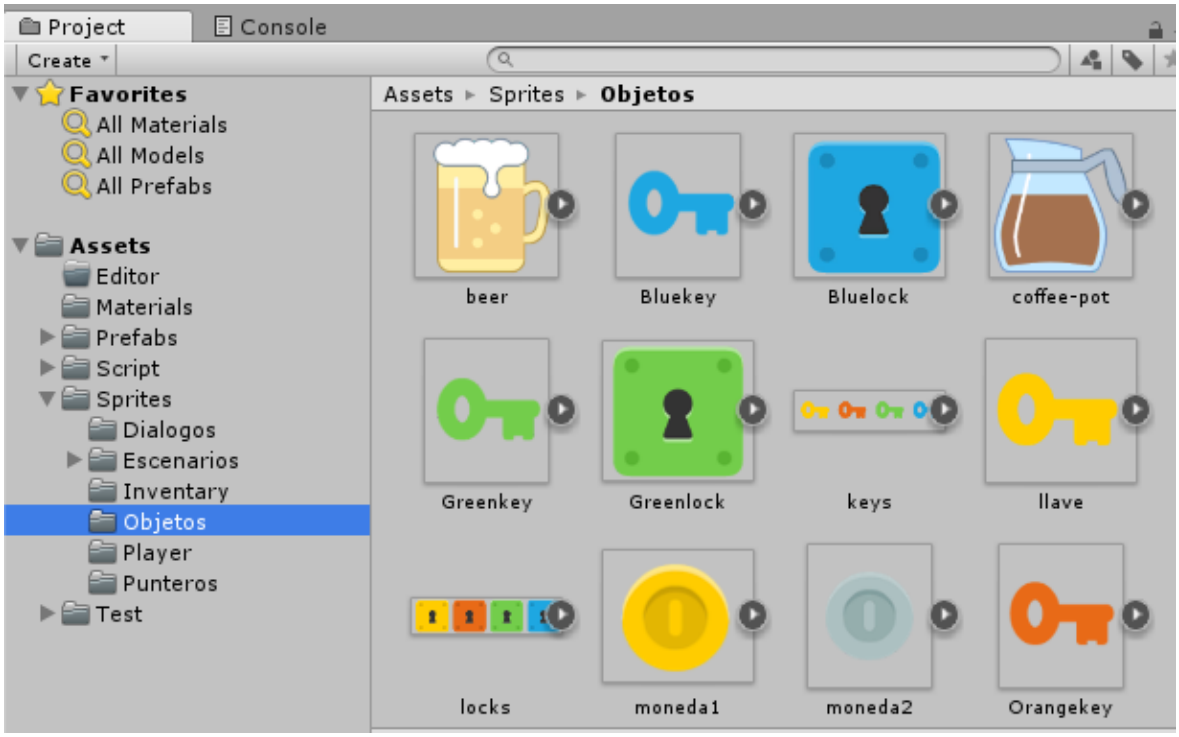

Figura AnexoB-18: Carpeta de Sprites de Objetos

- **Carpeta de los tests:** Aquí se encuentra la demo creada para probar los scripts y prefabs.

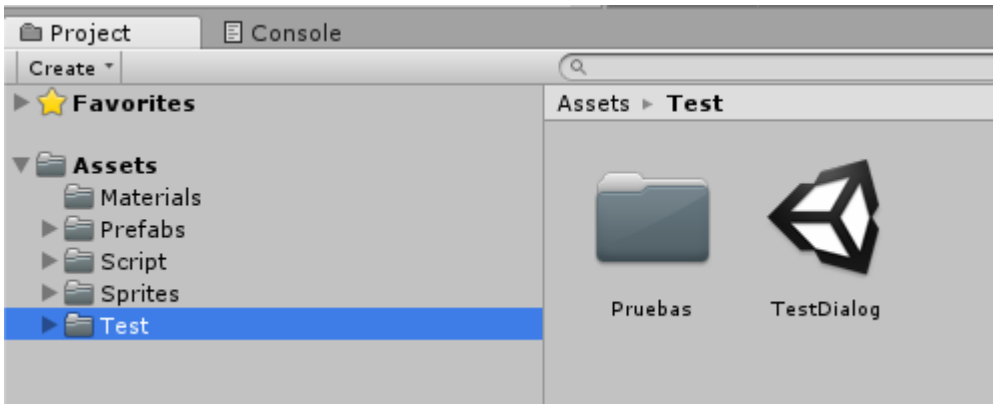

Figura AnexoB-19: Carpeta de Test

Estructura y uso de los prefabs del Asset:

- **Prefab de Cursor.** Está formado por el script:

  - **ControlCursor:** se puede modificar el tamaño del puntero del ratón y asignar dos iconos distintos para detectar interacciones durante el juego.

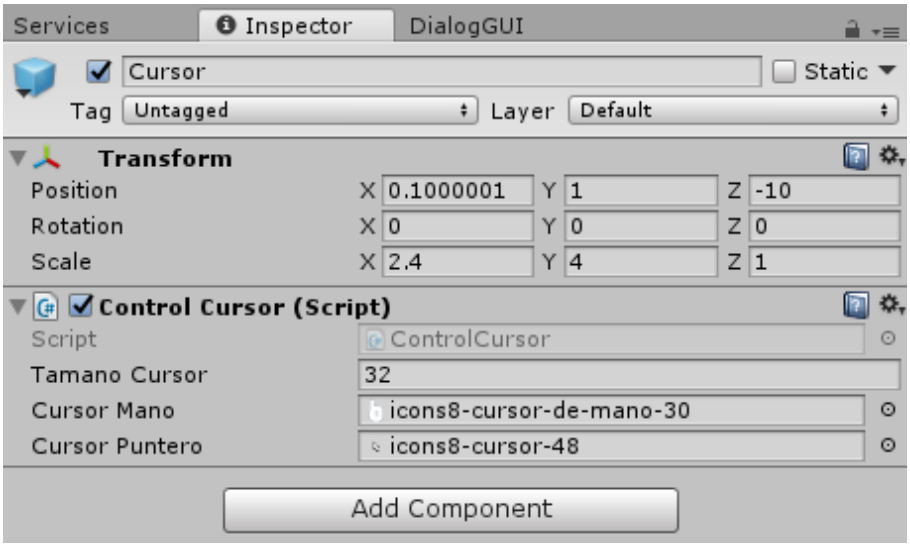

Figura AnexoB-20: Prefab de Cursor

- **Prefab de Dialog:**

  - **CanvasDialog:** está formado por la siguiente estructura, el conjunto de diálogos y un panel con texto. En este prefab solo habría que editar los diálogos por separado.

  - **DialogX.** Está formado por los siguientes scripts:

    DialogControl: hay que indicar un texto, los botones y la escena.

    DialogManager: hay que indicar la clave de dialog y el propio archivo de diálogo. Además, se pueden añadir objetos de eventos para el jugador.

  A continuación, se mostrarán dos diálogos, uno sin objetos y otro con objetos:

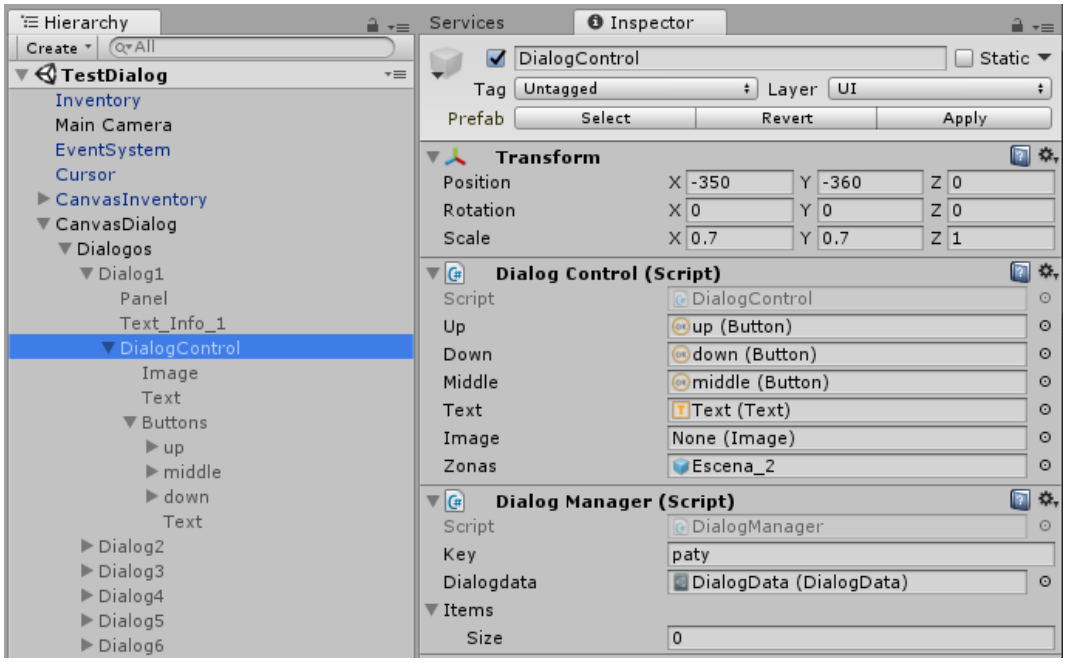

Figura AnexoB-21: Diálogo sin objetos

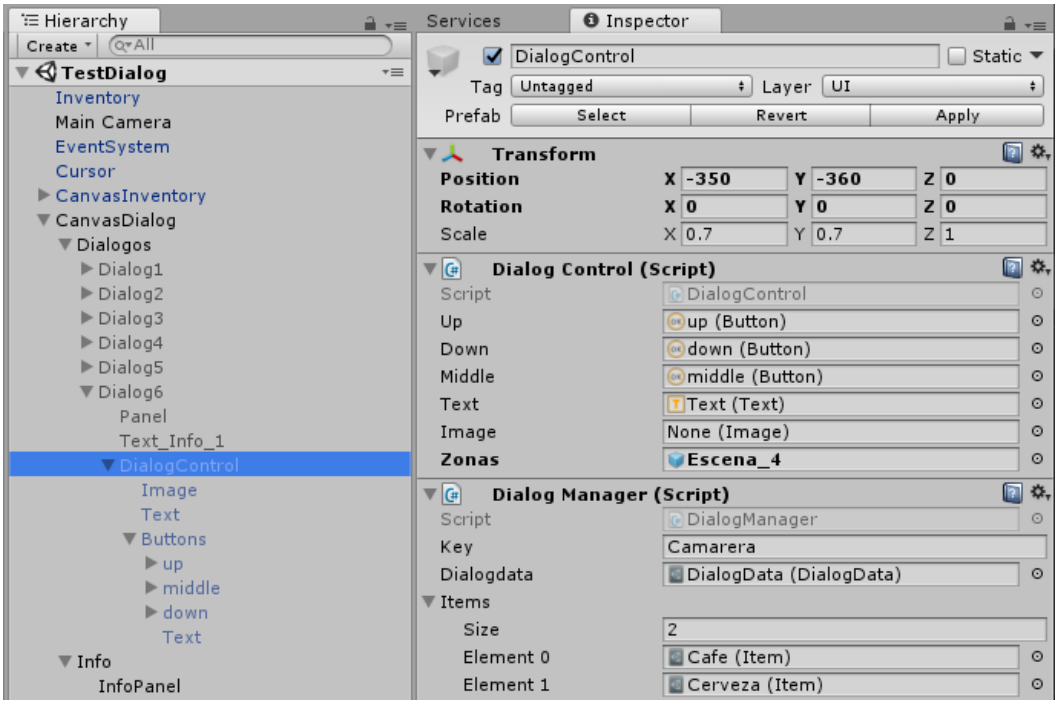

Figura AnexoB-22: Diálogo con Objetos

- **Prefab de Escenas:**

  - **Escena:** Este prefab está formado por todas las escenas donde cada una de ellas puede tener diferentes tipos de interacciones y objetos.

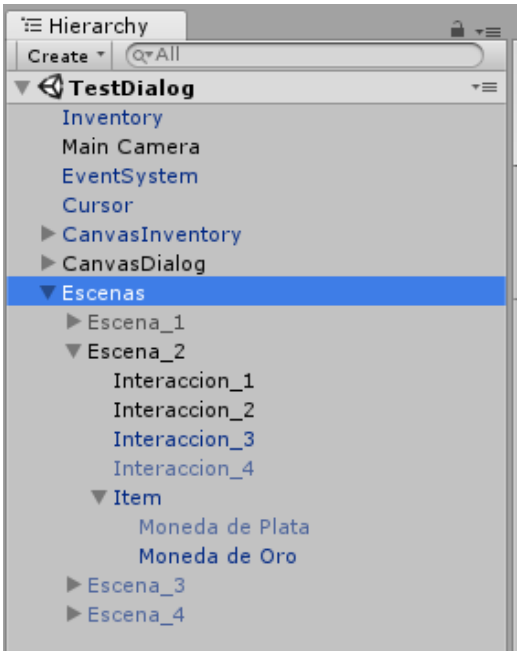

Figura AnexoB-23: DialogData.asset (2)

- **Interacción de diálogo:**

  ZonaInteracción: hay que indicar los prefabs de Dialog y Diálogos que se van a usar y un texto indicativo de la interacción.

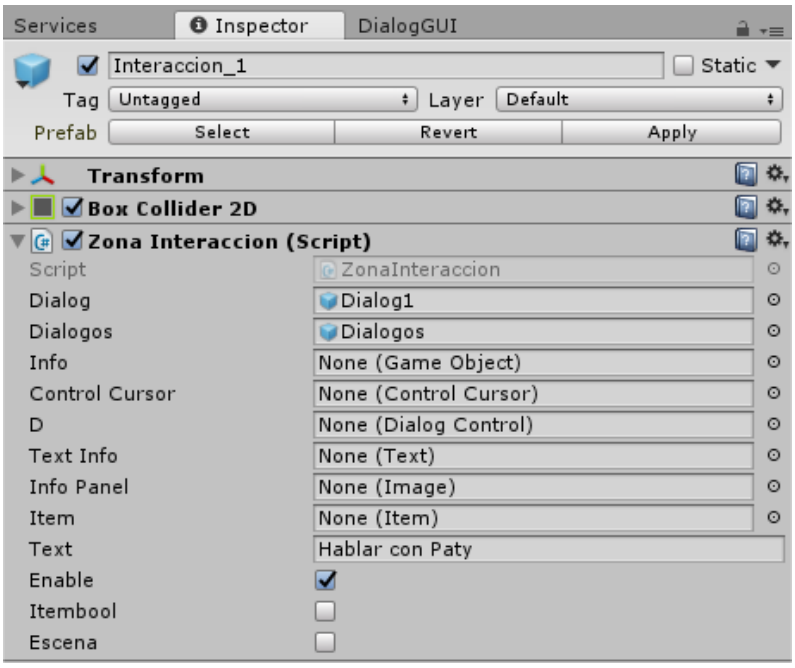

Figura AnexoB-24: Interacción de diálogo

- **Interacción de cambio de escena sin evento:**

  EscenaInteracción: hay que indicar el prefab de Diálogos y un texto indicativo de la interacción.

  CambiarEscenas: hay que indicar la escena a la que queramos cambiar y no marcar el campo Ev.

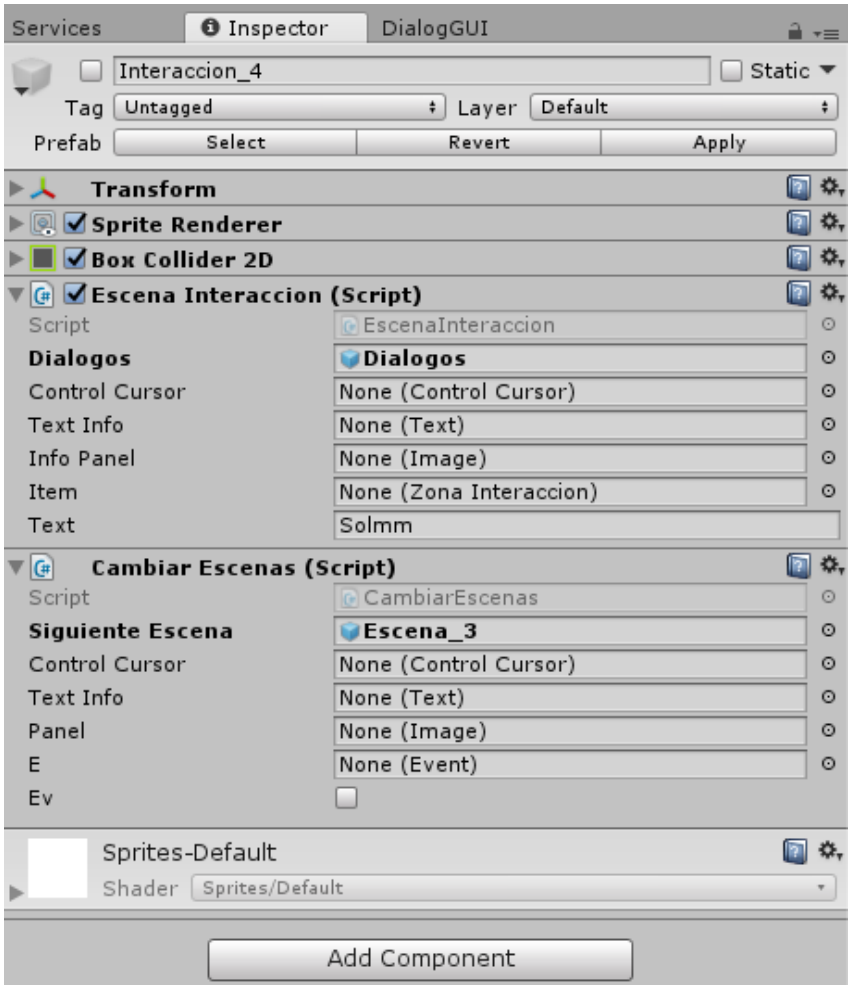

Figura AnexoB-25: Interacción de cambio de escena sin evento

- **Interacción de cambio de escena con evento:**

  ZonaInteracción: hay que indicar los prefabs de Diálogos y Dialog_X, un texto indicativo de la interacción y marcar el campo Escena.

  CambiarEscenas: hay que indicar la escena a la que queramos cambiar, se selecciona en el campo E un evento en forma de interacción y se marca el campo Ev.

  Event: se marca el campo Active y se elige el tamaño de la lista de objetos para el evento y el nombre de los objetos.

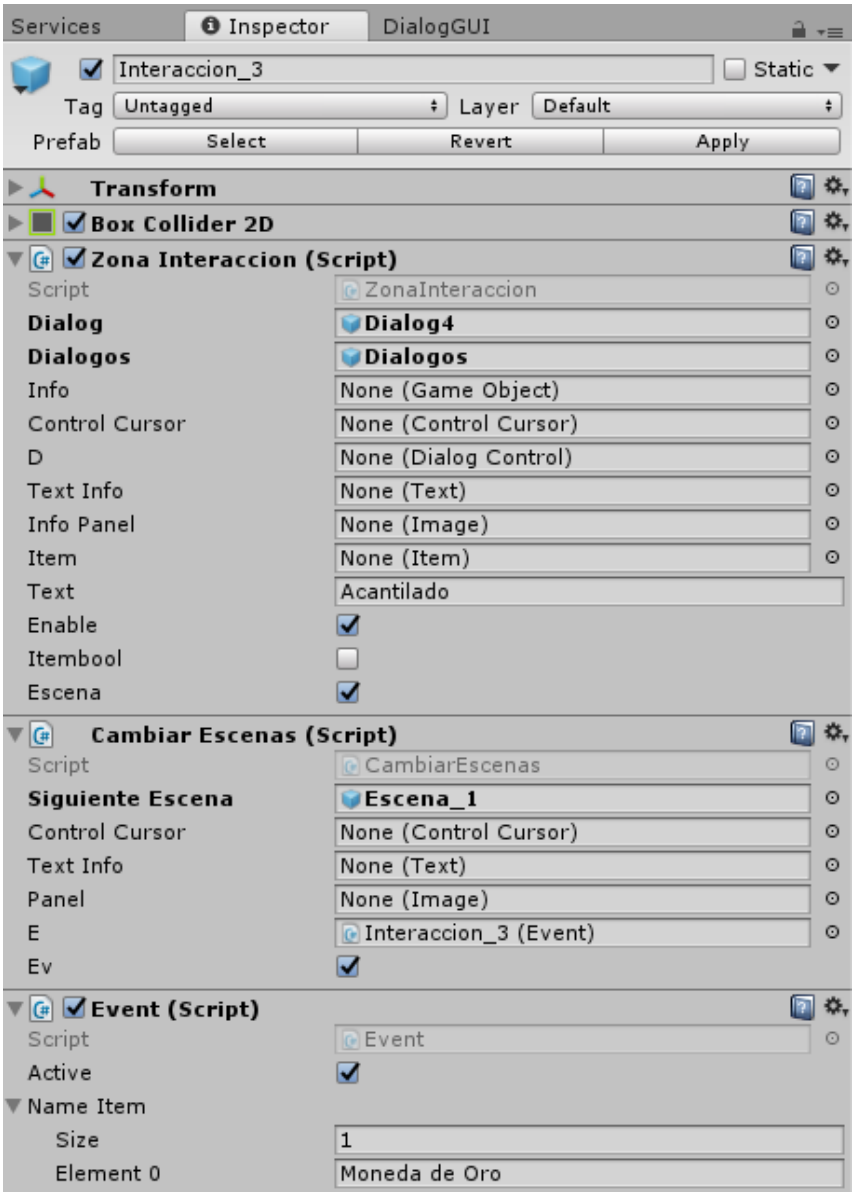

Figura AnexoB-26: Interacción de cambio de escena con evento

- **Prefab de Inventory:**

  - **Inventory:** este prefab instancia la clase Inventory con el patrón Singleton.
    Se puede establecer el número de espacios del inventario y su tamaño irá aumentando a medida que se obtengan objetos.

    Cabe decir que, si se aumenta o disminuye el espacio de objetos, también hay que añadir o quitar slots del gameObject “CanvasInventory”.

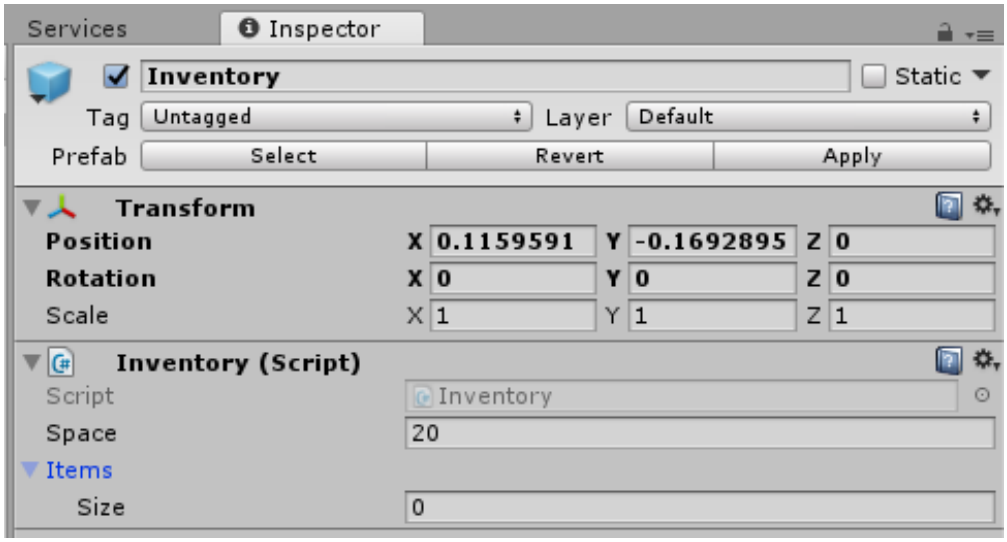

Figura AnexoB-27: Prefab Inventor

- **CanvasInventory:** este prefab abarca toda la interfaz del inventario, y su script InventoryUI necesita los siguientes gameObject y opciones:

    - Grupo de slots llamado ItemsParent.
    - Inventory, que es el prefab que hemos mencionado antes.
    - Prefabs de los Dialogos, incluido en el prefab de CanvasDialog.
    - Prefabs de las Escenas.
    - Marcamos el campo Dialog, para que no se produzcan errores con los diálogos.

    Además, dentro del gameObject de Inventory se encontrará la imagen, los slots, los textos de título y la descripción que conforman el inventario.

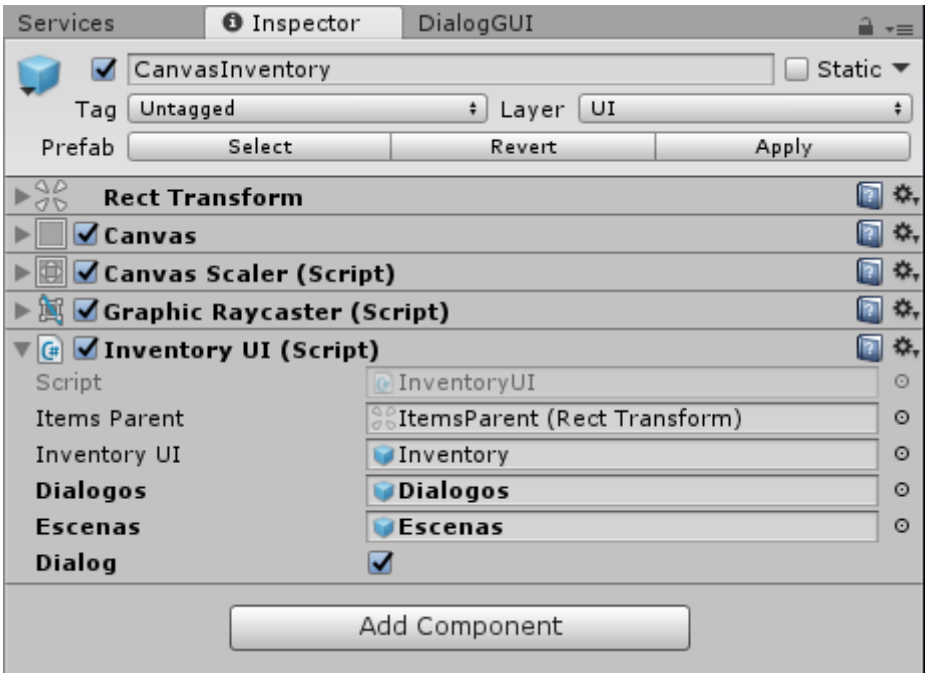

Figura AnexoB-28: Prefab CanvasInventory

- **Slot:** este prefab conforma un hueco del inventario formado por el script InventorySlot, que a su vez tiene la imagen del slot y un botón para eliminar el objeto.

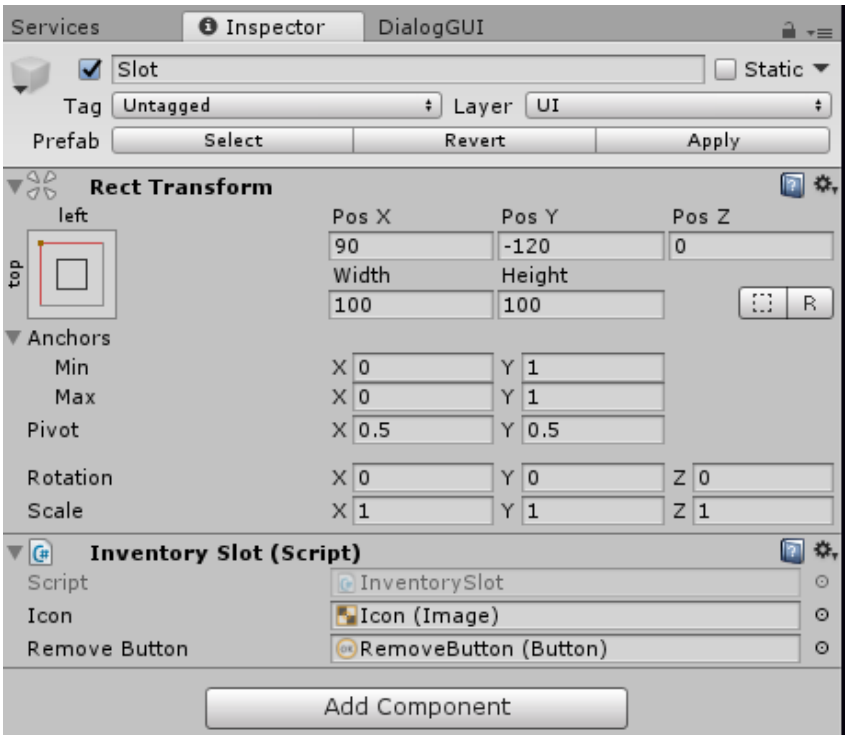

Figura AnexoB-29: Prefab Slot

Además, está formado por otros gameObject que son el ItemButton y el RemoveButton, y realizan la apertura de la descripción y borrado del objeto respectivamente.

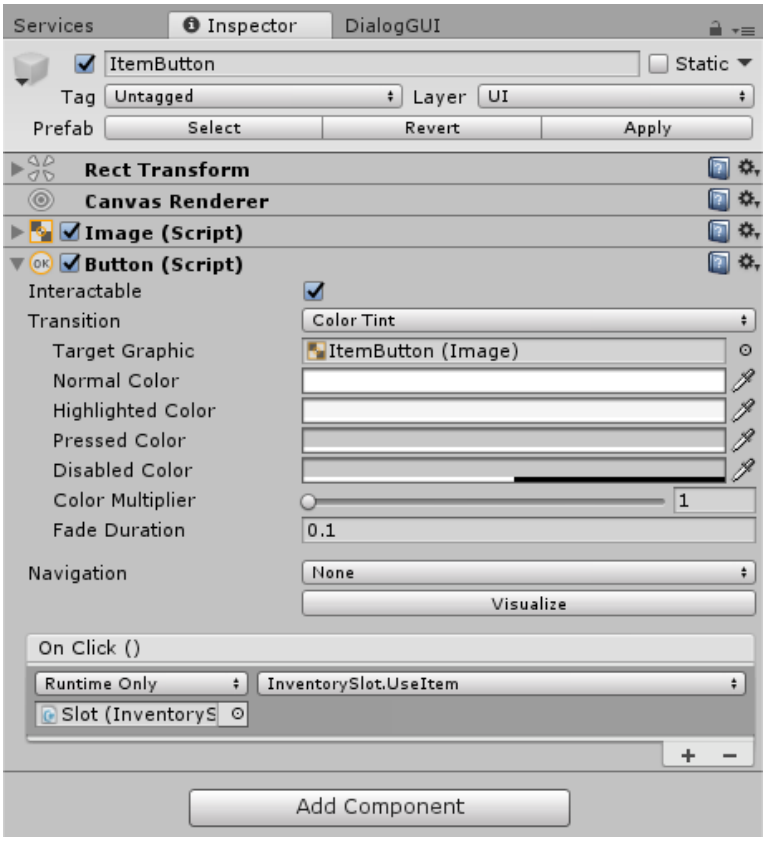

Figura AnexoB-30: Prefab ItemButton

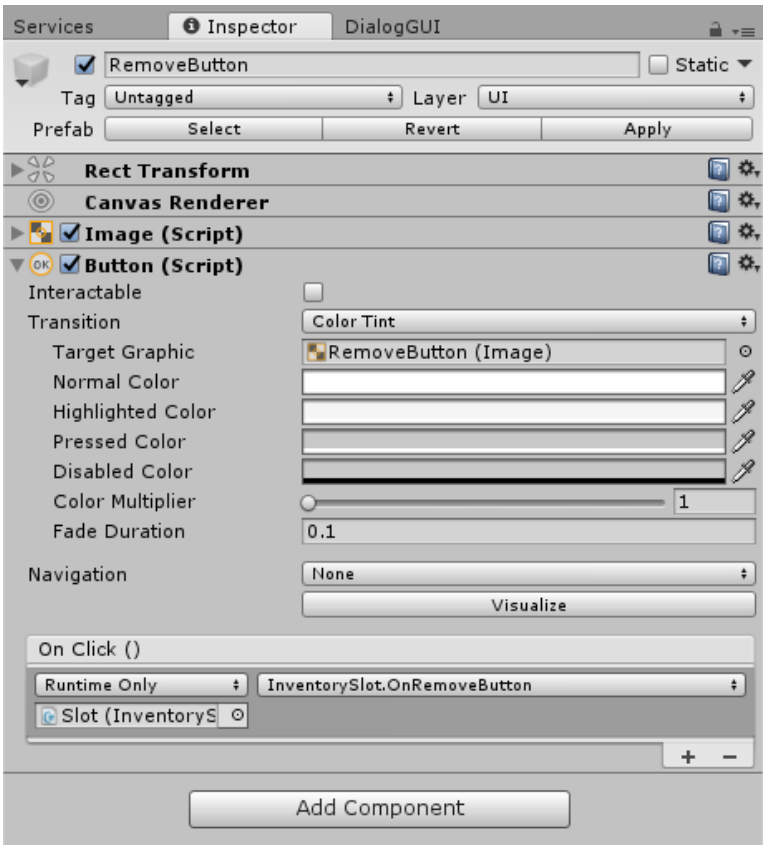

Figura AnexoB-31: Prefab RemoveButton

C Anexo Imágenes de los test

Estas son las imágenes de las pruebas que se han llevado a cabo para comprobar el funcionamiento de los scripts y de los assets.

Para obtener estas imágenes se creó una pequeña demo jugable con la cual se probaba la funcionalidad creada para el proyecto.

Pruebas de diálogos:

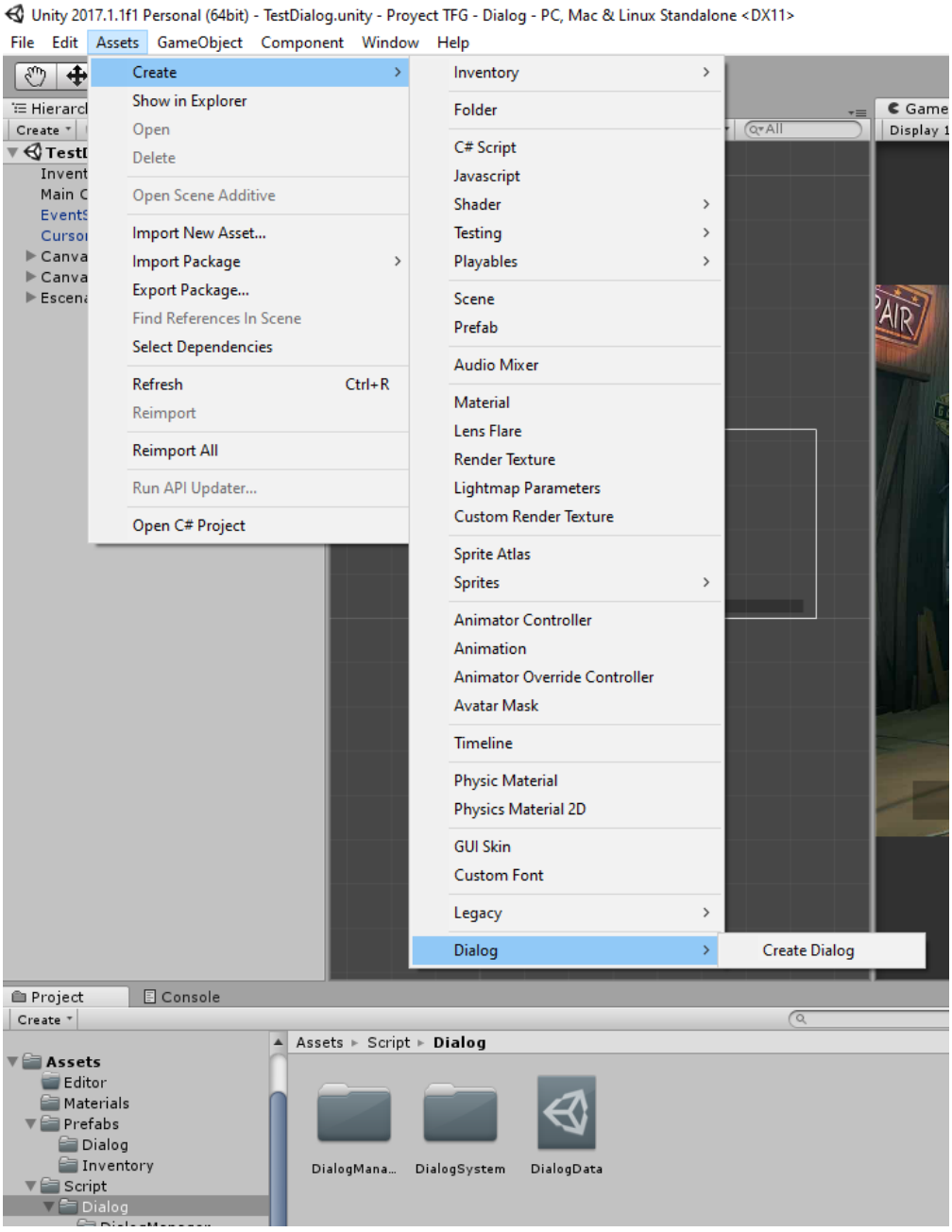

Figura AnexoC-32: Creación del diálogo

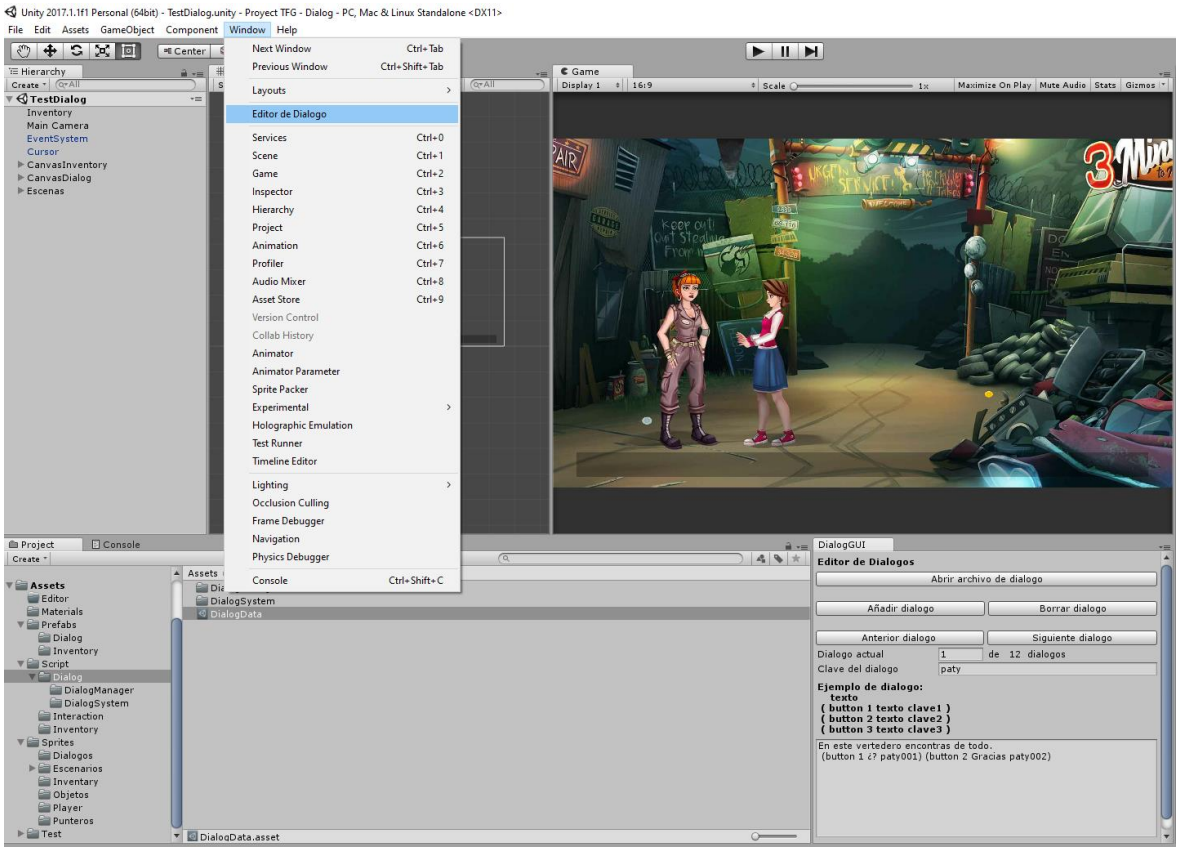

Figura AnexoC-33: Diálogo UI

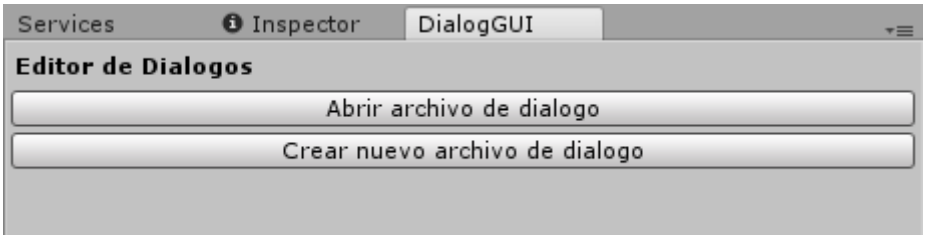

Figura AnexoC-34: Crear nuevo diálogo o abrir uno ya existente diálogo

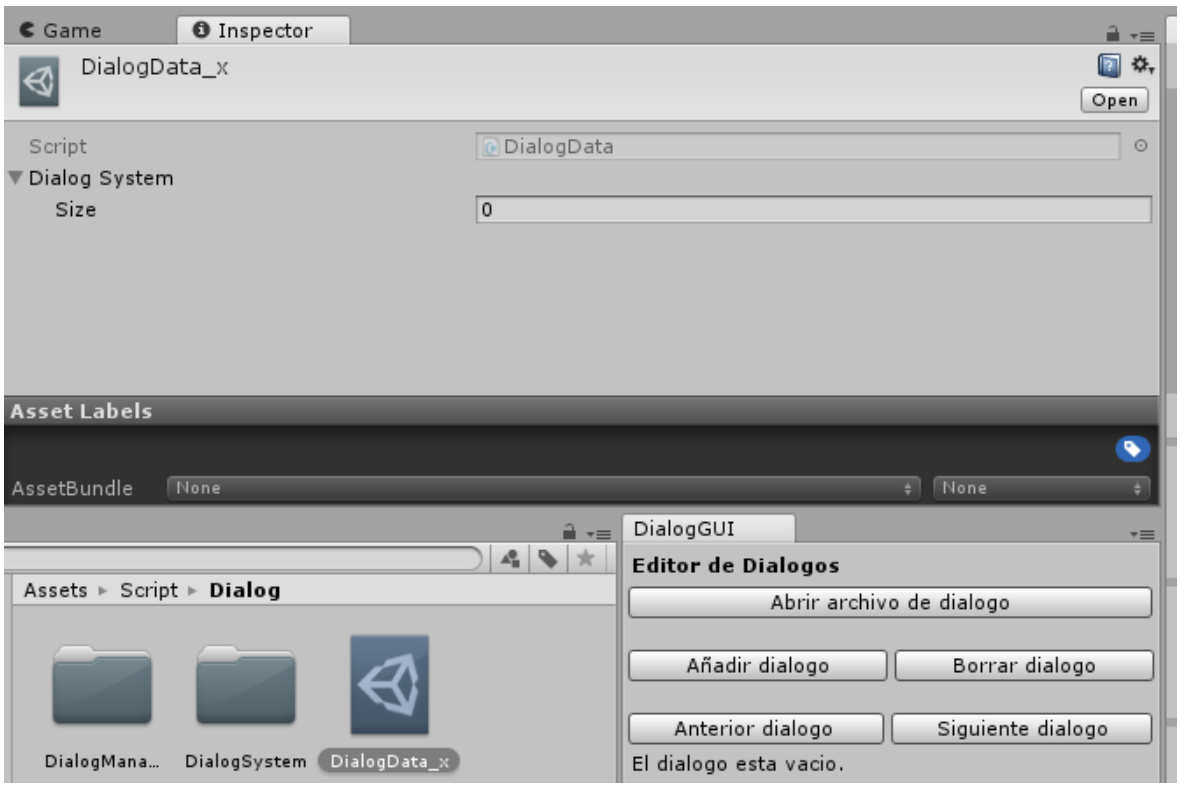

Figura AnexoC-35: Datos de un diálogo nuevo

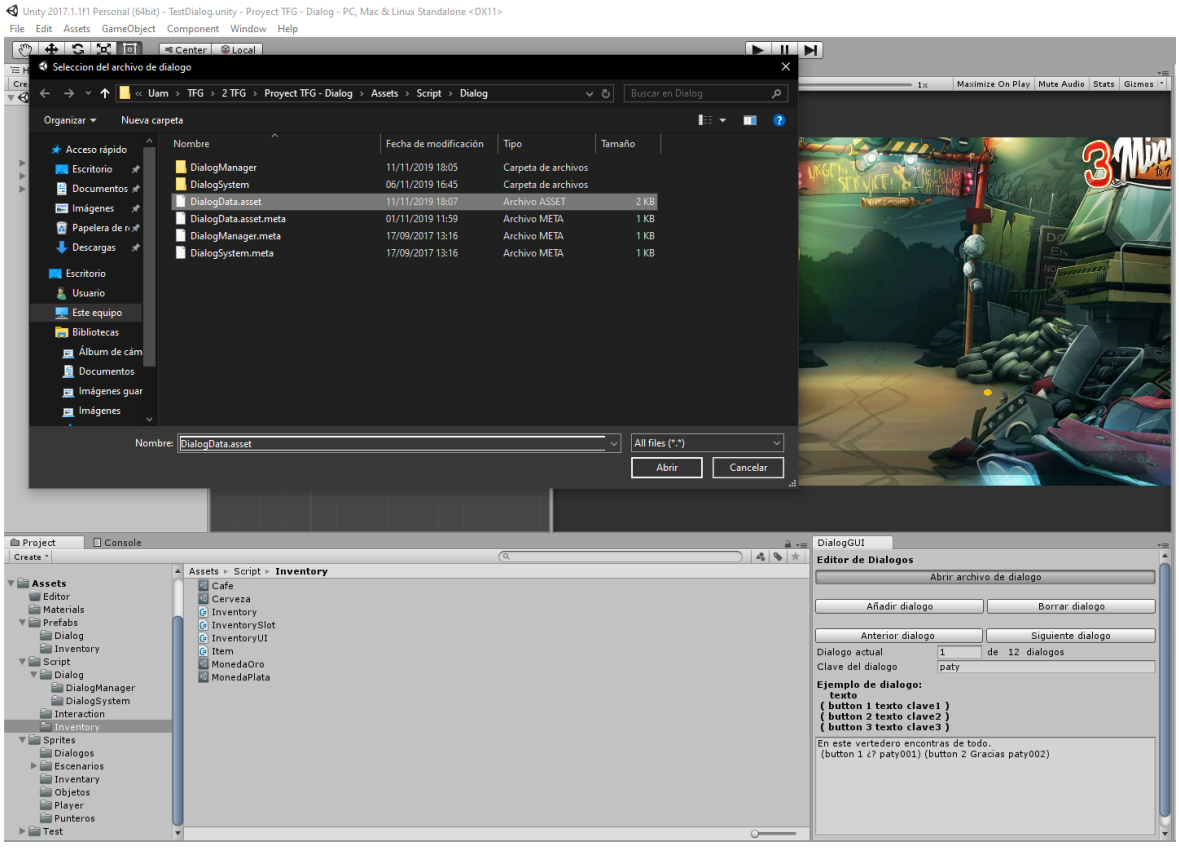

Figura AnexoC-36: Apertura de un diálogo

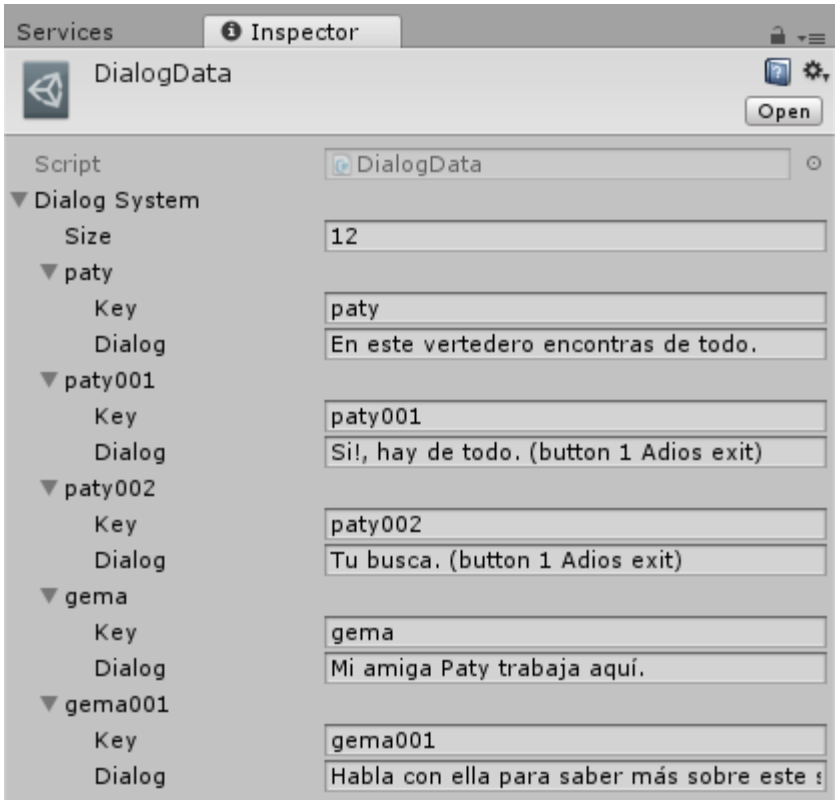

Figura AnexoC-37: Datos de un diálogo

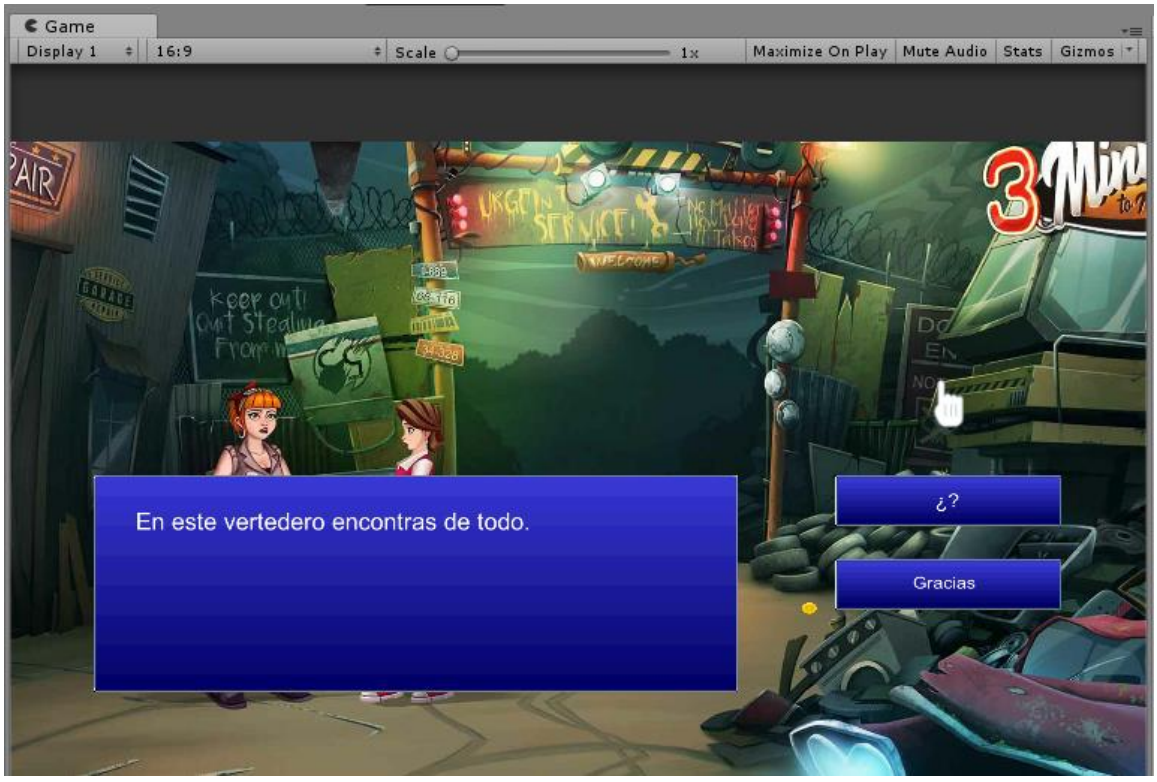

Figura AnexoC-38: Ejemplo de diálogo

Pruebas de interacciones:

Figura AnexoC-39: Interacción de diálogo

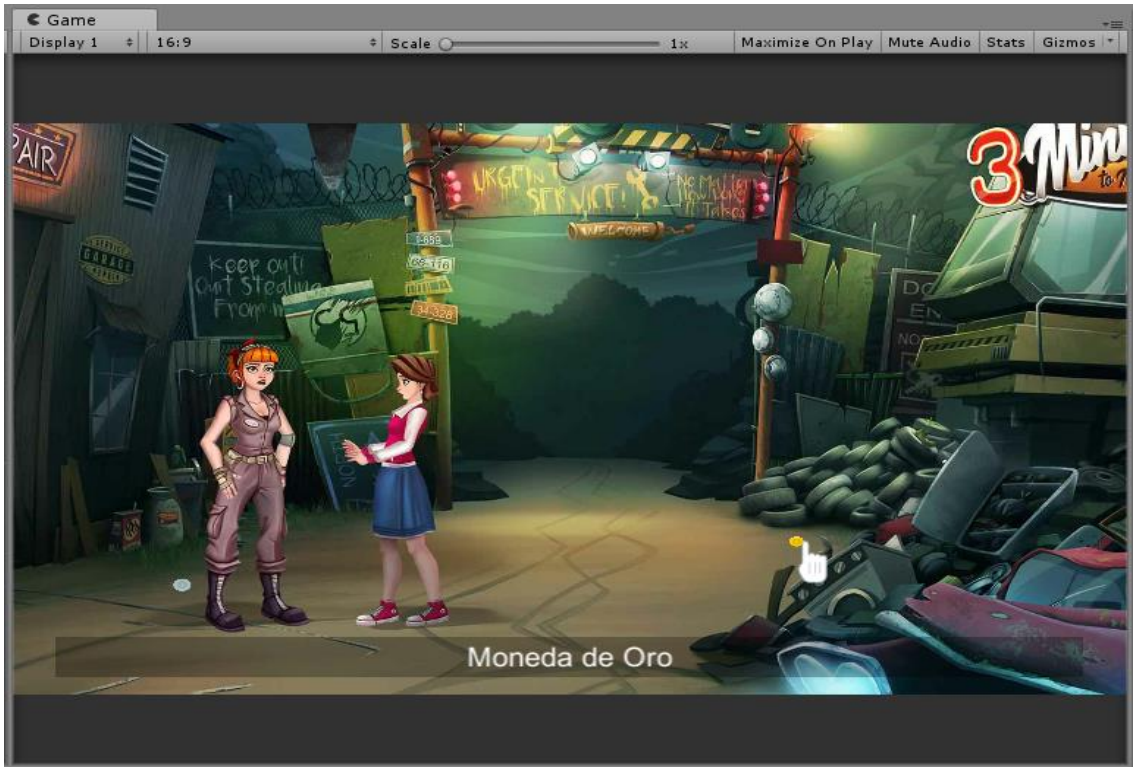

Figura AnexoC-40: Interacción de objeto

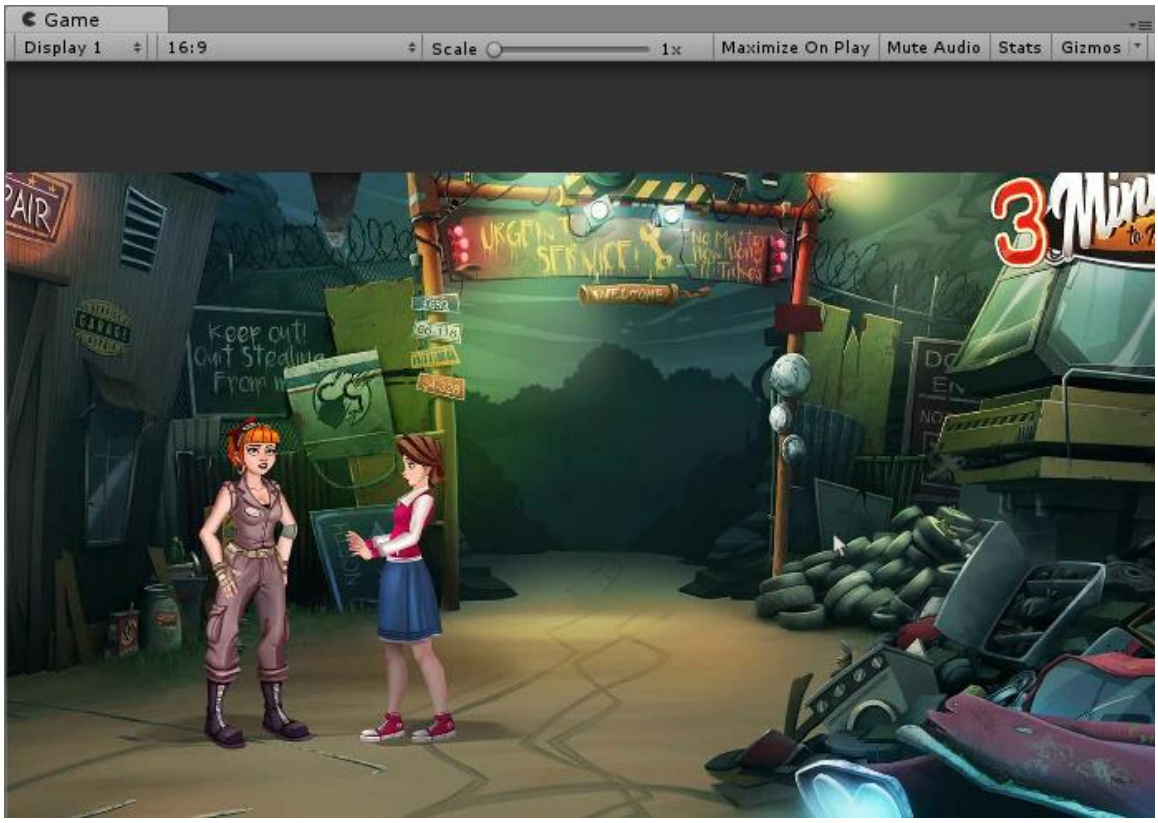

Figura AnexoC-41: Interacción de objeto cogido

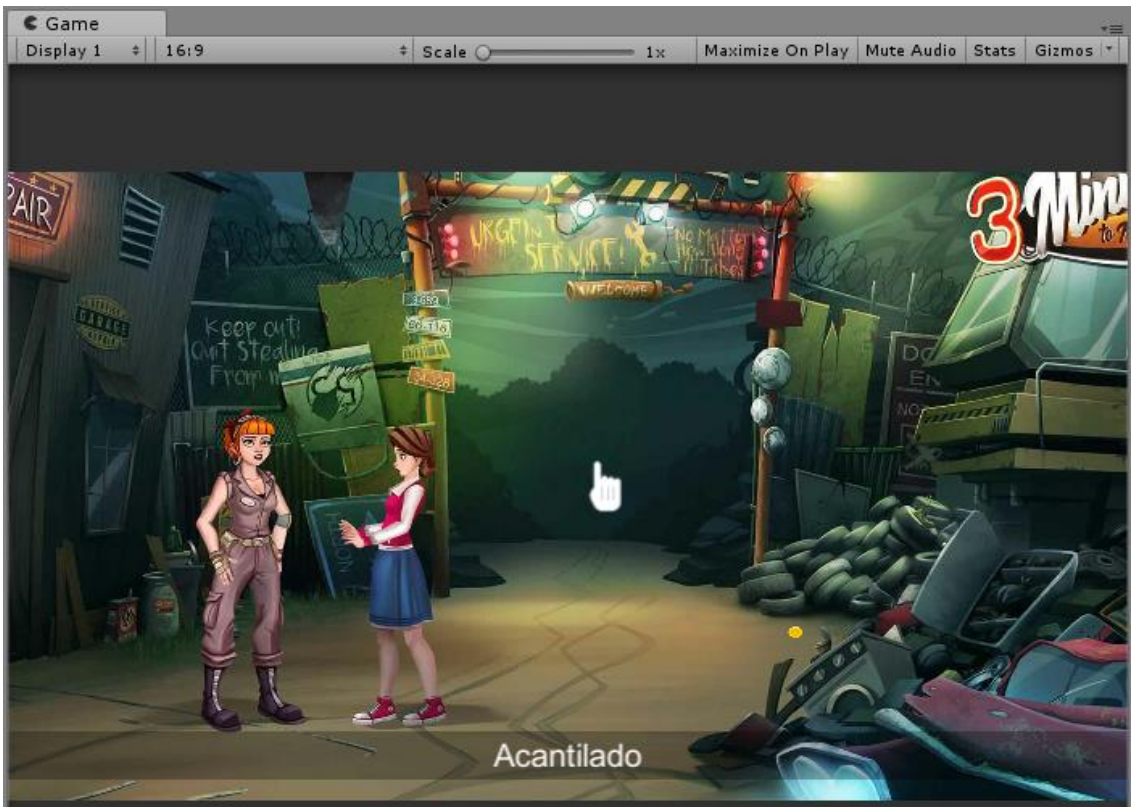

Figura AnexoC-42: Interacción de cambio de escena

Pruebas de inventario:

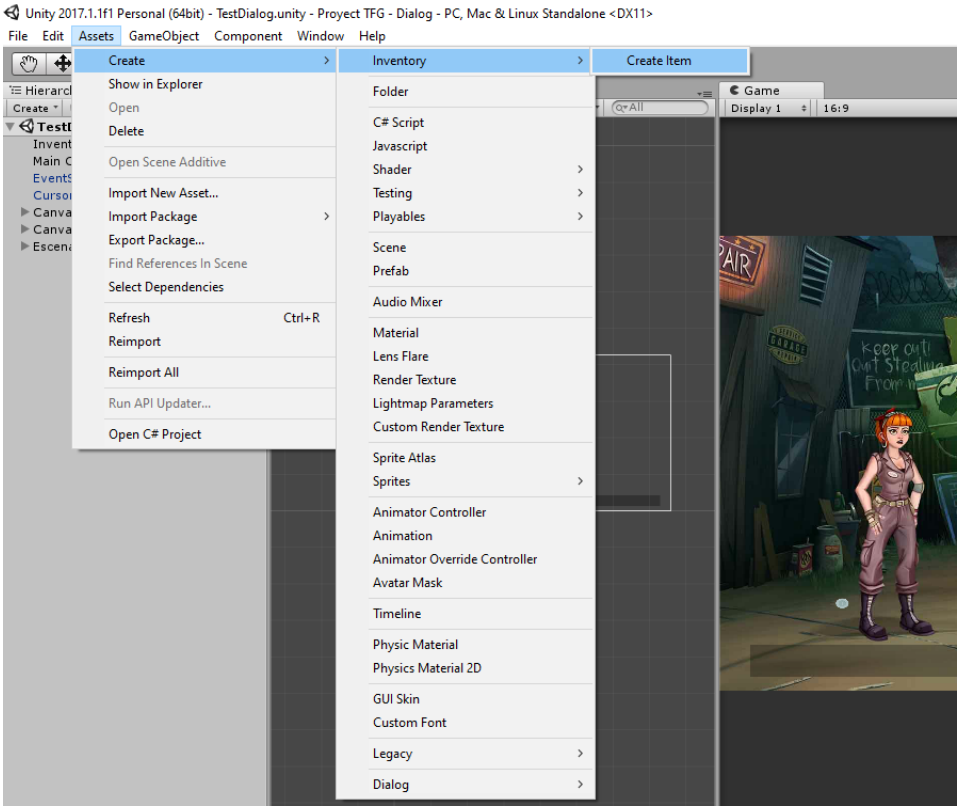

Figura AnexoC-43: Create Item

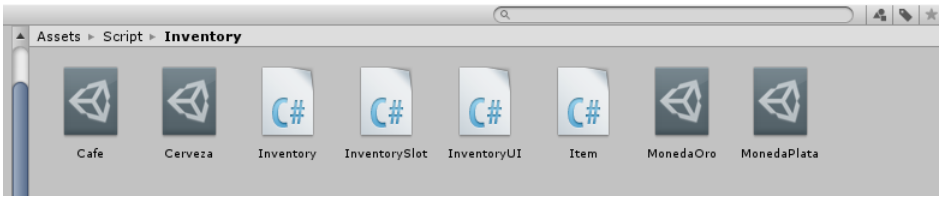

Figura AnexoC-44: Items

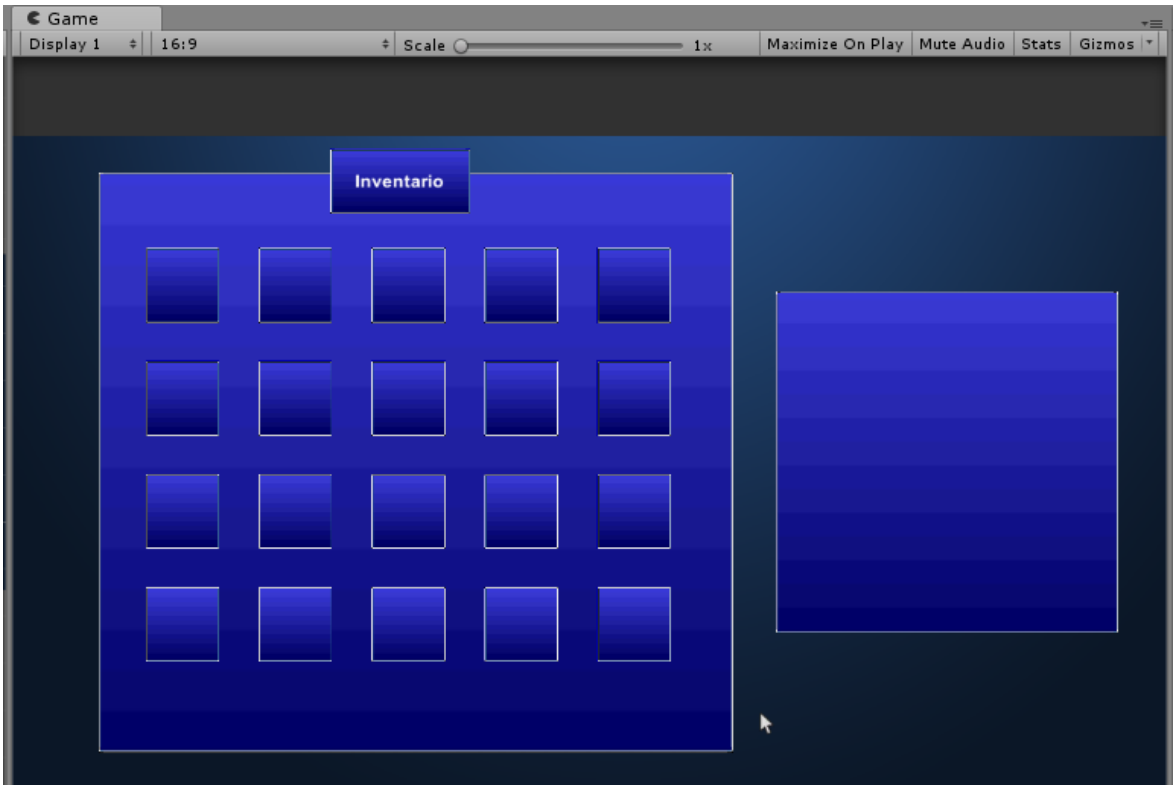

Figura AnexoC-45: Inventario vacío

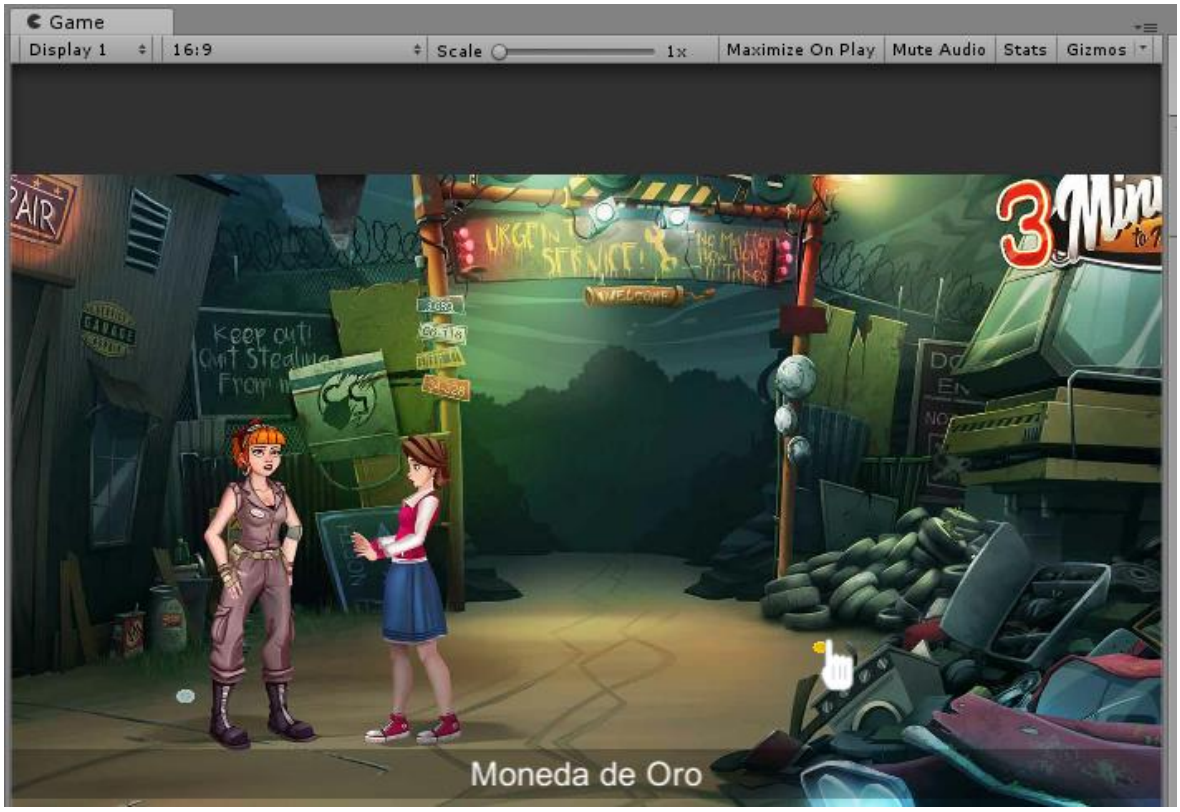

Figura AnexoC-46: Item moneda de oro

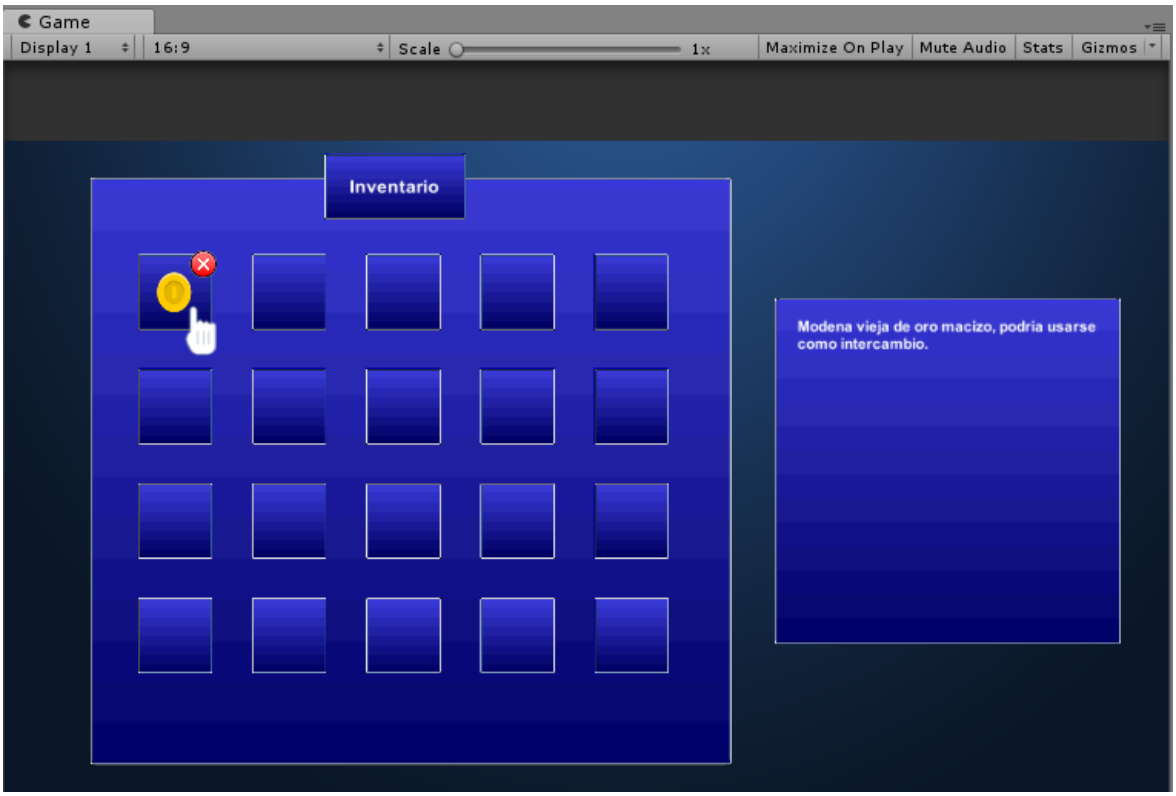

Figura AnexoC-47: Inventario con ítem y descripción

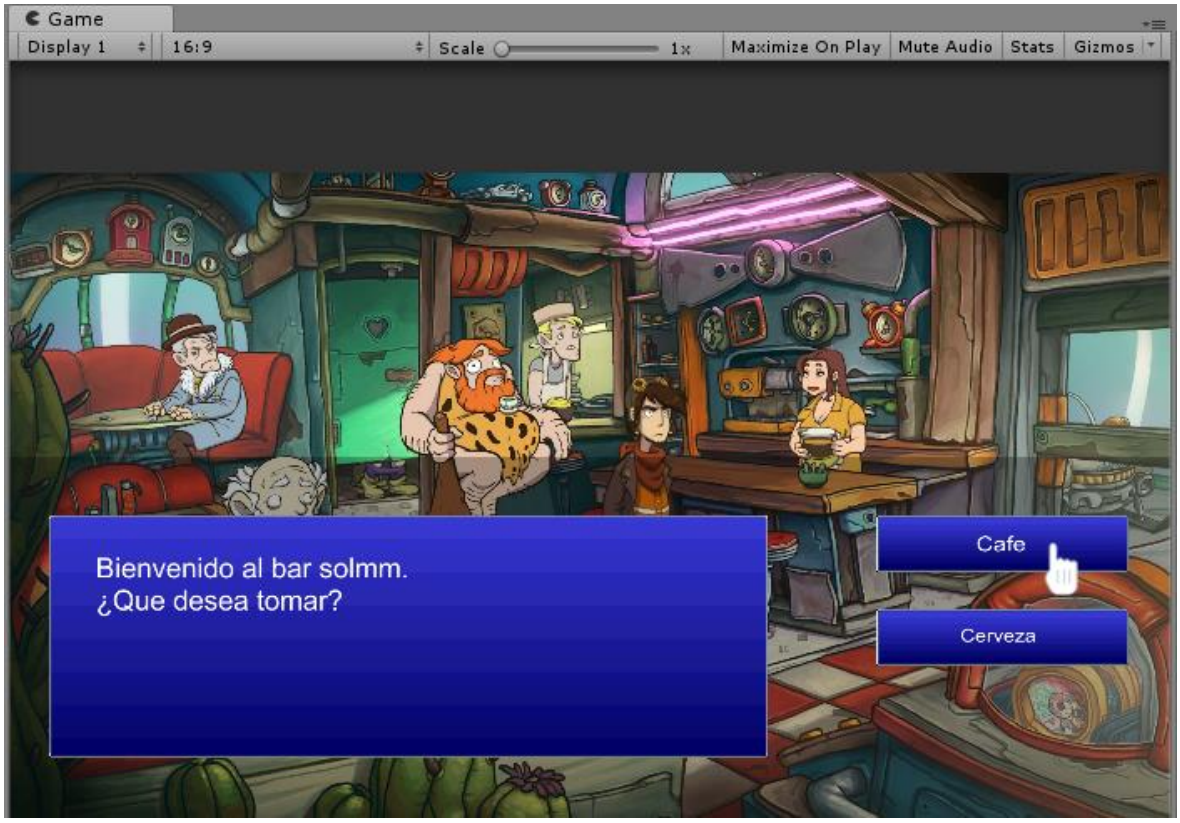

Figura AnexoC-48: Item por diálogo

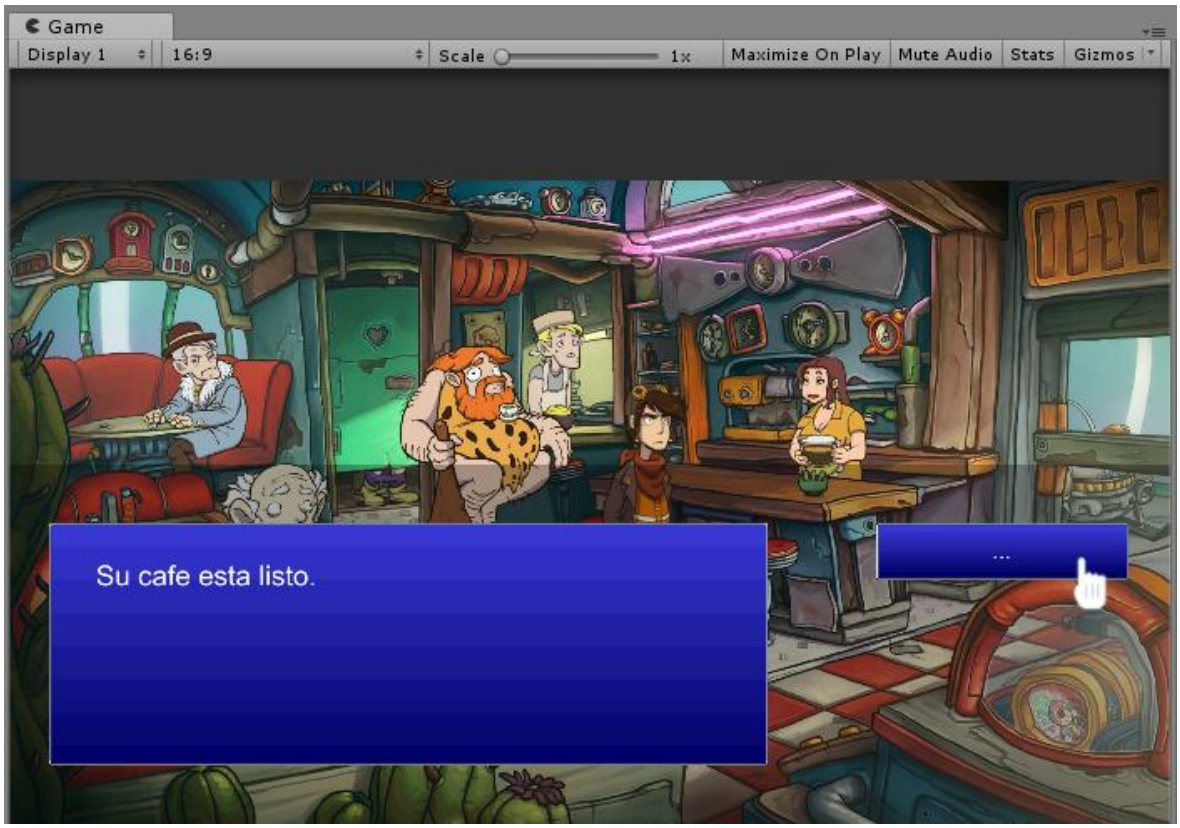

Figura AnexoC-49: Item por diálogo 2

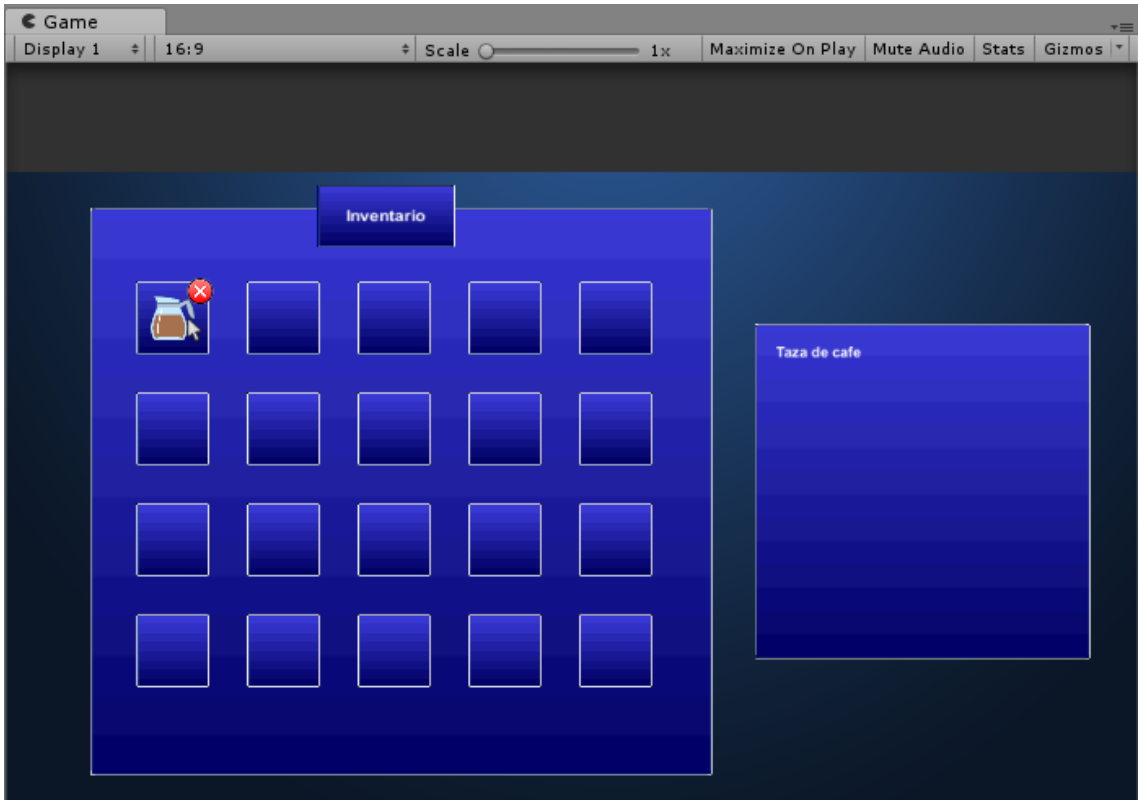

Figura AnexoC-50: Item por diálogo 3

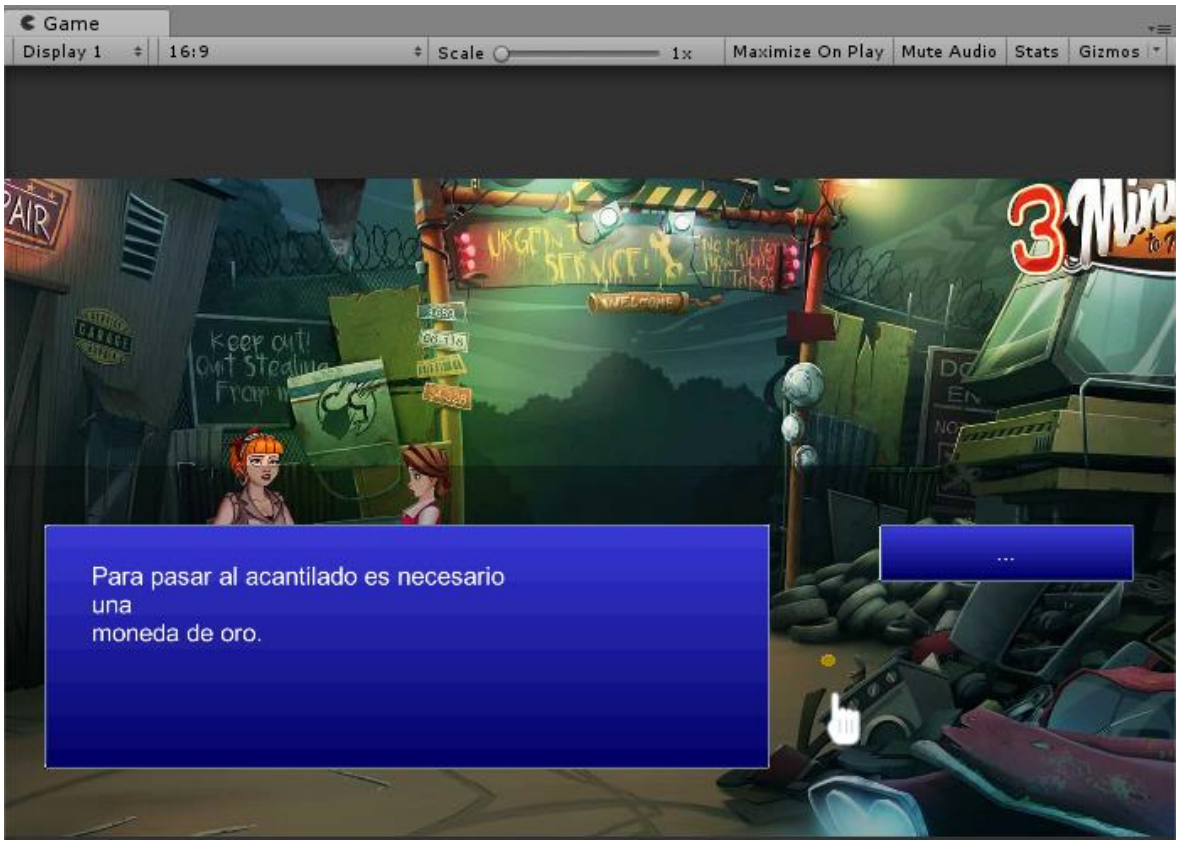

Figura AnexoC-51: Bloqueo de cambio de escena
